VadaTech AMC520

User's Manual

July 18, 2014

Version 2.2.2

vadatech inc.

THE POWER OF VISION ™

Notice

While reasonable efforts have been made to assure the accuracy of this document, VadaTech, Inc. assumes no liability resulting from any omissions in this document or from the use of the information obtained herein. VadaTech reserves the right to revise this document and to make changes periodically and the content hereof without obligation of VadaTech to notify any person of such revision or changes.

Electronic versions of this material may be read online, downloaded for personal use, or referenced in another document as a URL to the VadaTech Incorporated Web site. The text itself may not be published commercially in print or electronic form, edited, translated, or otherwise altered without the permission of VadaTech, Inc.

It is possible that this publication may contain reference to or information about VadaTech products (machines and programs), programming, or services that are not available in your country. Such references or information must not be construed to mean that VadaTech intends to announce such products, programming, or services in your country.

Revision History

Doc Rev	Description of Change	Revision Date
0.0.1	Initial version	2/7/2012
1.0.0	Documentation completed	4/4/2012
2.0.0	Updated entire document to match Rev B board.	9/15/2012
2.0.1	Updated document to match Rev C board and software.	10/14/2013
2.1.0	Clarified the size and configurations of the QDRII+ SRAM. Added section describing how to modify the hardware to convert channels between OpAmp and Magnetic after purchase.	3/18/2014
2.2.0	Added CREN:USERIOTEST and CREN:RTMTEST registers fields and associated description to enable the front panel User I/O and RTM Data/Clock loopback diagnostic tests. Added board photo. Provided an improved block diagram. Added Front Panel User I/O connector description. Added capture and capture_all script description to simplify ADC data capture process.	7/16/2014
2.2.1	Expanded the DACCTRL:FIXED_GENx fields and added the DACFD register to allow arbitrary fixed data value to be presented to the DAC channels.	7/17/2014
2.2.2	Changes to add DAC loop-through from ADC channels feature: Added description of the DAC loop-through feature. Added DAC MMCM status bits to BRDSTATUS register. Expanded the DACCTRL:FIXED_GENx fields and dropped the FIXED_ portion of the name to add values for each ADC channel.	7/18/2014

Table of Contents

Figures

Tables

1 Document Overview

This document describes the AMC520 board including the FPGA reference design, host-side device driver/tool, and configuration of the MMC microcontroller. It also describes how to go about using/customizing the FPGA reference design for customer specific needs.

Further FPGA/software development is generally expected to be performed at the customer's site to add any additional application-specific functionality to the AMC520 board. The reference design FPGA/software implementation is provided as an example and proof-of-concept but is not considered a formal baseline for the customer's application and it may change at any time.

This document describes the Rev C and later version of the board. Prior board versions are not supported.

1.1 Applicable Products

- VadaTech AMC520 (Virtex-6)
- Related product:
    - VadaTech AMC514 (Virtex-6 - AMC520 derives from this design)

1.2 Document References

- AMC520 FPGA Pin-out/Design Diagrams (found in AMC520 VHDL Sources release)
- VadaTech AMC520 Datasheet (http://www.vadatech.com)
- PICMG® AMC.0 R2.0 AdvancedMC Mezzanine Module (http://www.picmg.org)
- PICMG® AMC.1 R2.0 AdvancedMC PCI Express and AS (http://www.picmg.org)
- PICMG® AMC.2 R1.0 AdvancedMC Ethernet (http://www.picmg.org)
- PICMG® uTCA.4 R1.0 Enhancements for Rear I/O and Precision Timing (http://www.picmg.org)
- Xilinx Virtex-6 Datasheets and User's Guides (http://www.xilinx.com/support/documentation/virtex-6.htm)
- Xilinx Virtex-6 Integrated Block for PCI Express (PCIe) Documentation (http://www.xilinx.com/products/ipcenter/V6_PCI_Express_Block.htm)
- Xilinx Virtex-6 Embedded Tri-Mode Ethernet MAC Wrapper Documentation (http://www.xilinx.com/products/ipcenter/V6_Embedded_TEMAC_Wrapper.htm)
- Xilinx XAUI Documentation (http://www.xilinx.com/products/ipcenter/XAUI.htm)
- Xilinx Memory Interface Generator Documentation (http://www.xilinx.com/products/ipcenter/MIG.htm)

1.3 Acronyms Used in this Document

Acronym	Description
A/D	Analog to Digital Converter
ADC	Analog to Digital Converter
AMC	Advanced Mezzanine Card
BAR	Base Address Register
BIST	Built-In Self Test
BPI	Byte Peripheral Interface
CGND	Chassis Ground
CLK	Clock
CPU	Central Processing Unit
D/A	Digital to Analog Converter
DAC	Digital to Analog Converter
DDR3	Dual Data Rate 3 SDRAM
DIP	Dual In-line Package
DMUX	De-multiplexer
DR	Data Ready
FPGA	Field Programmable Gate Array
FRU	Field Replaceable Unit
GbE	Gigabit Ethernet
GND	Signal Ground
GTX	Virtex-6 Gigabit Transceiver
ioctl	Input/Output/Control
IP	Intellectual Property / Internet Protocol
IPMI	Intelligent Platform Management Interface
JSM	JTAG Switch Module
JTAG	Joint Test Action Group
LED	Light Emitting Diode
LVCMOS	Low-Voltage Complementary Metal Oxide Semiconductor
LVDS	Low Voltage Differential Signaling
MAC	Media Access Controller
MB	Megabyte (2^20 bytes)
MIG	Memory Interface Generator
M-LVDS	Multi-point Low Voltage Differential Signaling
mmap	Memory Map
MMC	Module Management Controller
MMIO	Memory Mapped Input/Output
MUX	Multiplexer
n.c.	No connection
PCIe	Peripheral Component Interconnect Express
PHY	Physical Layer Device
PICMG	PCI Industrial Computer Manufacturer's Group
PIO	Programmed Input/Output
SDRAM	Synchronous Dynamic Random Access Memory
SERDES	Serializer/Deserializer
SGMII	Serial Gigabit Medium Independent Interface

TCLK	Telephony Clock
TRN	Transaction (layer of PCIe implementation)

Table 1: Acronyms

2 Hardware Overview

The AMC520 is a 10-channel A/D and 2-channel D/A card with on-board FPGA.

Figure 1: AMC520 board photo

It is in a double-wide Physics AMC form-factor which includes the following primary components (your ordering option may vary slightly):

- 5 – Dual-channel 16-bit 125 Msps A/Ds (AD9268) w/ magnetic or op-amp from RTM for a total of up to 10 ADC channels
- 1 – Dual-channel 16-bit 250Msps D/A (MAX5878) to RTM for a total of up to two DAC channels
- FPGA Block:
    - Xilinx Virtex-6 FF1759 FPGA with optional density and speed:
        - LX240T -1 or -2 (partially pinned FF1759 part)
        - LX365T -1 or -2 (partially pinned FF1759 part)
        - LX550T -1 or -2 (fully pinned FF1759 part)
        - SX475T -1 or -2 (fully pinned FF1759 part)
    - Option for QDRII+ SRAM
        - Channels: Single channel configuration
        - Size: 2Mbit x36 (8MB + parity) or 2Mbit x72 (16MB + parity)
        - Chips: One or two x36 chips of CY7C25652KV18-400 or equivalent
    - Programmable (defaulting to 300MHz and 400MHz) clock generators
    - Fixed 100MHz and 125MHz clock generators
    - Flexible ADC clock source selection

  - Backplane M-LVDS Clock/Trigger Transceivers (SN65MLVD080)
  - Backplane PCIe 2x4 or x8 connectivity w/ active repeaters (AMC Ports 4-11)
  - Backplane Dual 1000Base-X connectivity (AMC Ports 0 & 1)
  - Backplane Arbitrary SERDES connectivity (AMC Ports 2 & 3 and 12-15)
  - Front panel Dual SFP+ cages
  - 4 – User LEDs
  - 16 – Status LEDs
  - FPGA Configuration DONE LED
  - Front panel CLK IN / TRIG IN / TRIG OUT
  - 16 – Front panel Arbitrary USER I/O
  - Front panel FPGA JTAG (switchable to backplane JSM)
  - RS-232 console port
  - 32MB BPI Configuration Flash (JS28F256P30T or equivalent)
- AMC MMC controller w/ IPMI LEDs, hot swap handle, RS-232, etc

2.1 Block Diagram

A simplified block diagram is shown below:

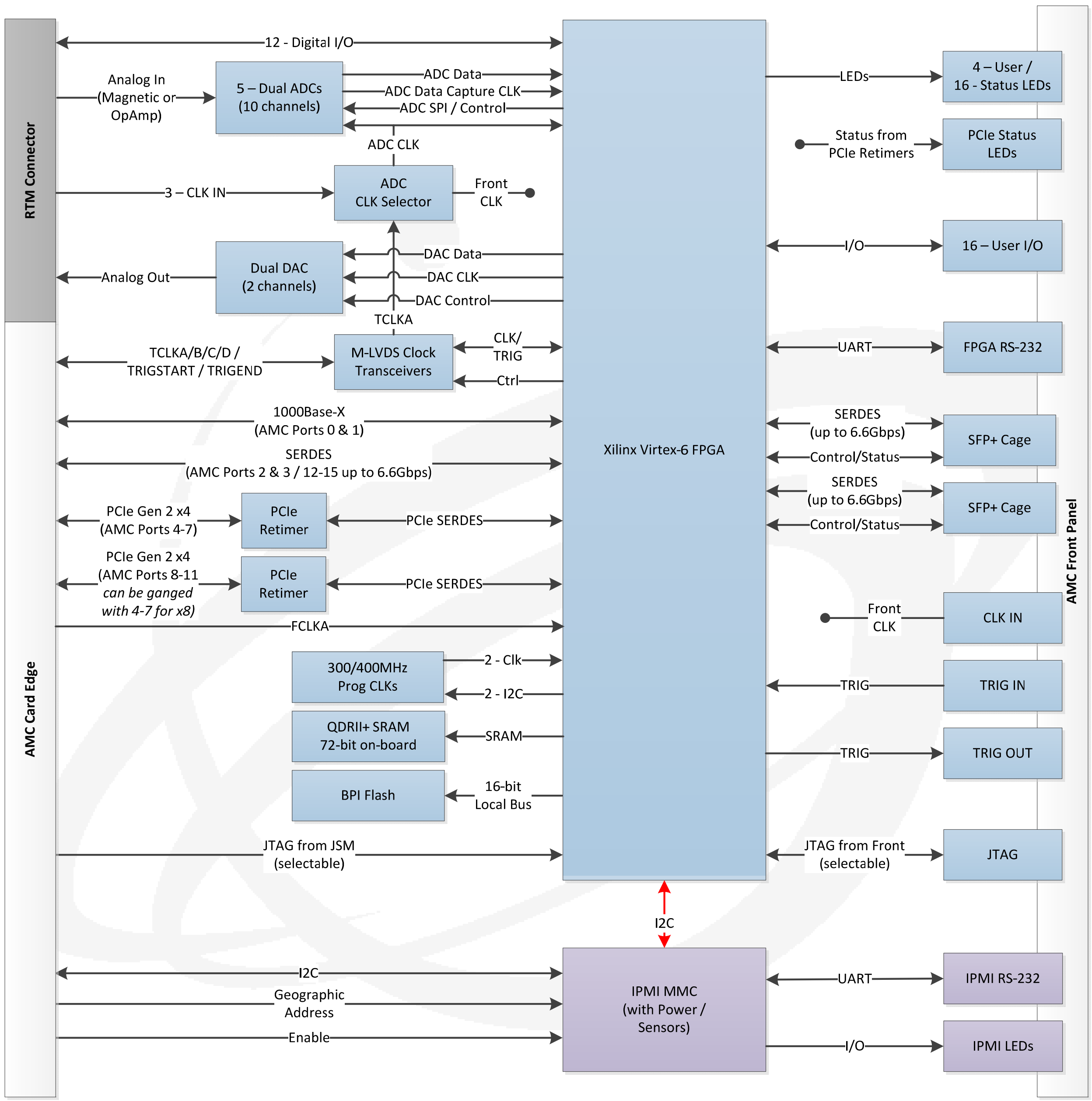


Figure 2: AMC520 simplified block diagram

2.2 Board Layout

The top-side layout of the card is shown below which includes user accessible jumpers:

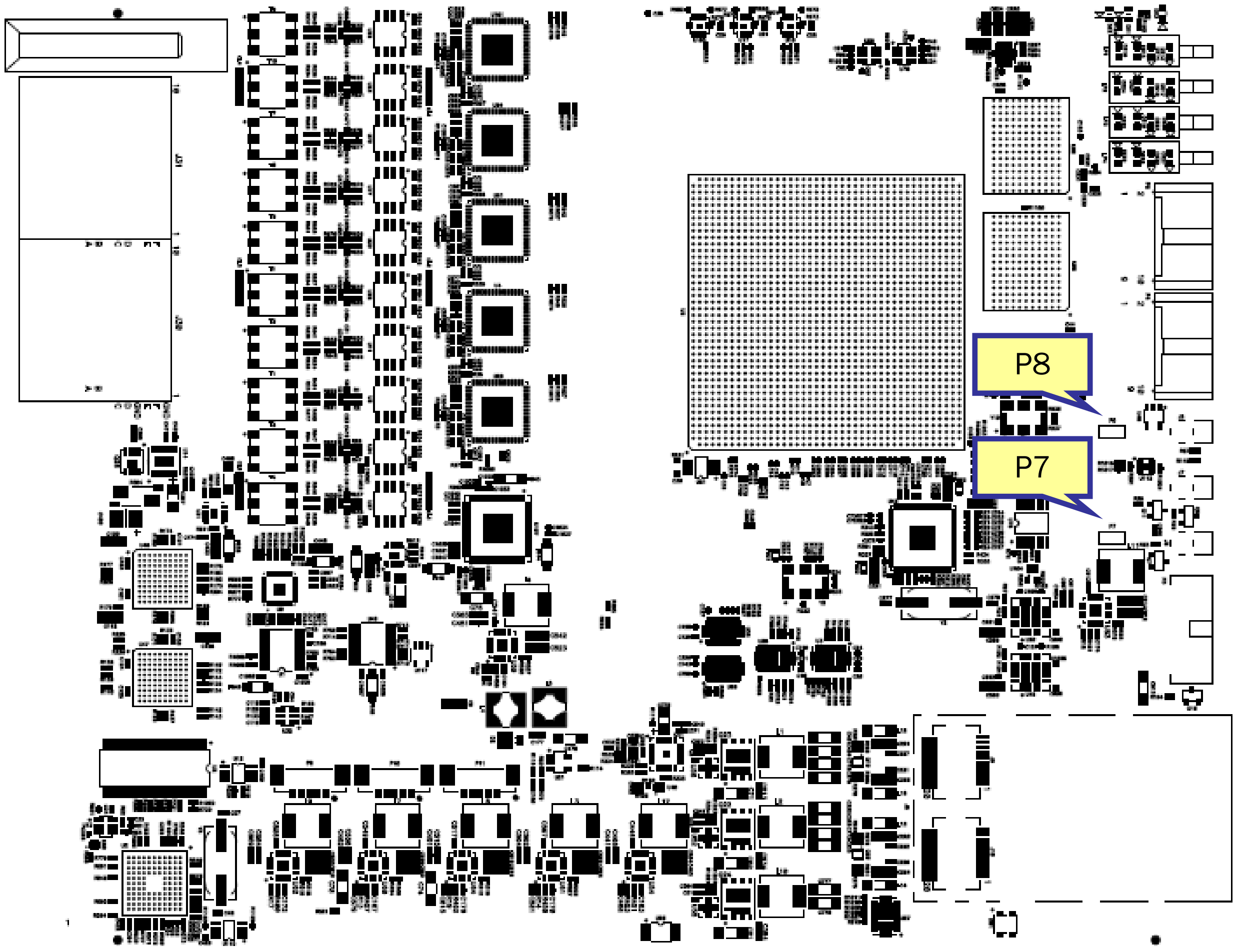


Figure 3: AMC520 top-side layout (front towards right)

The bottom-side layout of the AMC520 is shown below which includes configuration switches and test points:

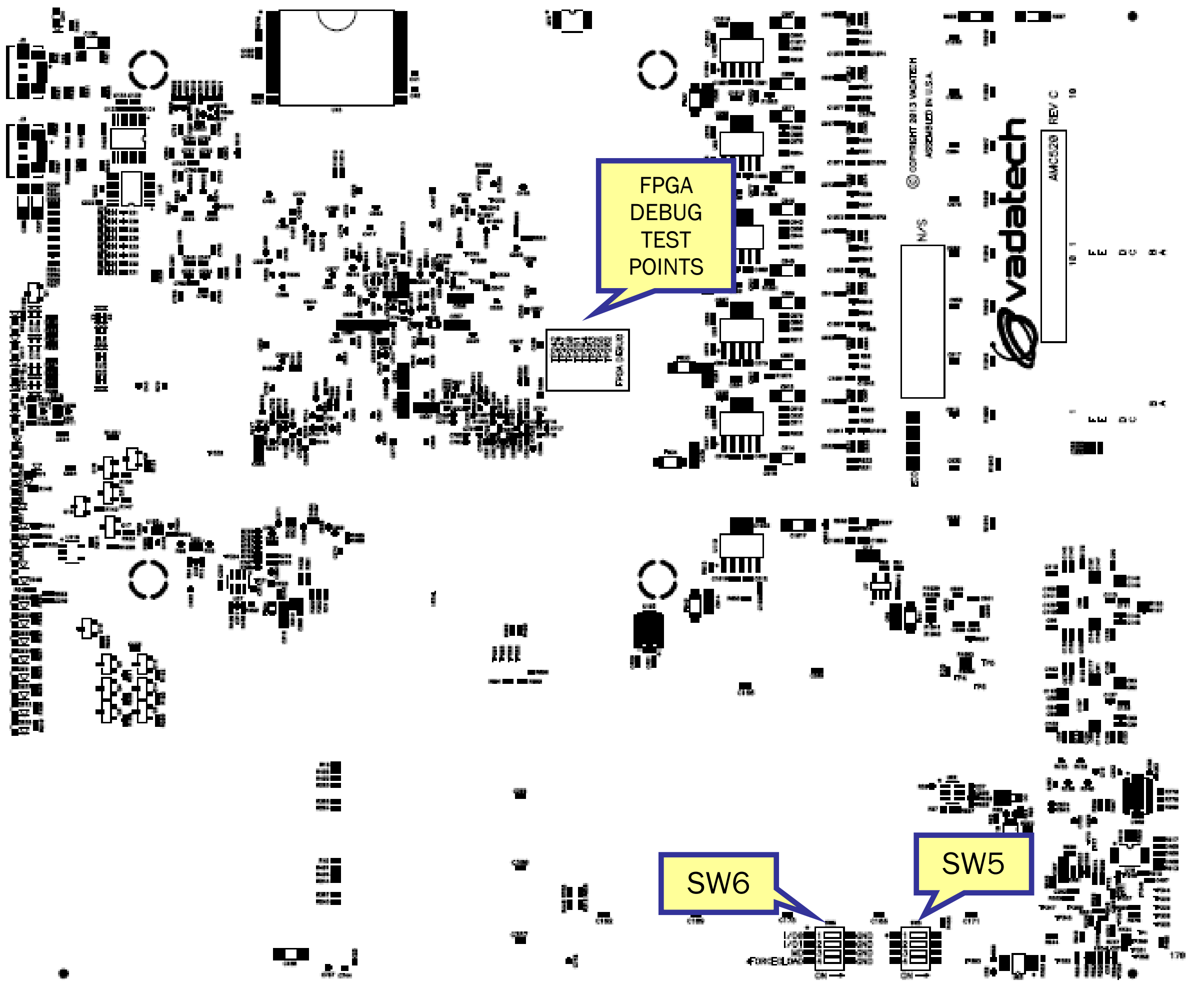


Figure 4: AMC520 bottom-side layout (front towards left)

2.3 On-board Switches

The card includes a set of DIP switches at **SW5** which control miscellaneous board functions as shown below:

SW5-	Off	On
1	Reserved [factory default]	Reserved (do not set)
2	Reserved [factory default]	Reserved (do not set)
3	Direct FPGA JTAG to front panel [factory default]	Direct FPGA JTAG to AMC connector (JSM)
4	Flash NOT write protected [factory default]	Flash write protected

Table 2: SW5 settings

The card includes a set of DIP switches at **SW6** which control the MMC microcontroller and are reserved for VadaTech use. These default to OFF-OFF-OFF-OFF, please do not change the setting without instruction from VadaTech.

2.4 On-board Headers/Jumpers

The jumper P7 enables 50ohm parallel termination for the TRIG IN front panel port.

Pin	Shunted	Open
1-2	50ohm termination	100Kohm weak pull-down

Table 3: P7 TRIG IN termination header

The jumper P8 enables 50ohm parallel termination for the CLK IN front panel port.

Pin	Shunted	Open
1-2	50ohm termination	100Kohm weak pull-down

Table 4: P8 CLK IN termination header

2.5 FPGA Debug Test Points

Test points are conveniently located within a silkscreen box on the back of the board to provide additional debug capability. They are listed in the order in which they appear within the box on the board.

Test Point	VHDL Name	FPGA Pin
TP262	DEBUG(0)	F12
TP255	DEBUG(1)	E12
TP253	DEBUG(2)	B16
TP254	DEBUG(3)	A16
TP251	DEBUG(4)	H15
TP250	DEBUG(5)	G14
TP247	DEBUG(6)	D16
TP242	DEBUG(7)	C16

Table 5: FPGA Debug Test Points

2.6 Front Panel Interfaces

The front panel of the AMC520 is shown below:

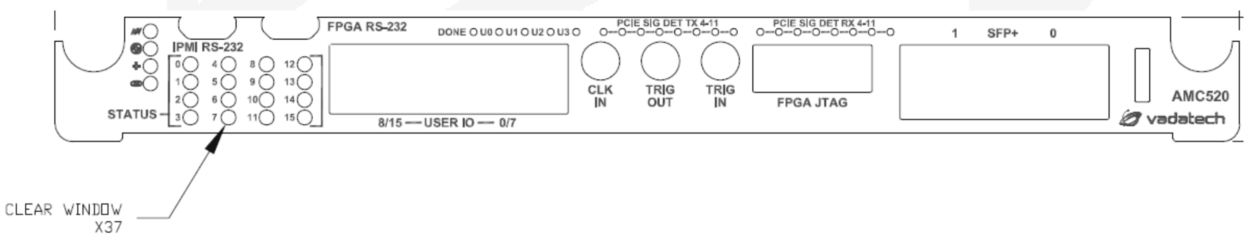


Figure 5: AMC520 front panel

2.6.1 Front Panel IPMI LEDs and Hot-Swap Handle

The front panel includes the standard AMC IPMI LEDs showing hot-swap status and general card health. The LEDs behave as follows:

LED	Off	On	Blink
Blue	Card active	OK to remove	Hot-swap/power transitioning
Red	No Fault	Payload Power Fault	n/a
Green	No payload power	Payload power OK	E-Keying failure
Amber	Normal	n/a	MMC flash writing

Table 6: AMC LED behavior

NOTE: The card should only be removed from a running carrier when the IPMI Blue LED is solid ON.

To insert the card, pull out the hot-swap handle until it stops. Insert the card into the carrier's guide rails and push on the front panel firmly until it is fully seated into the connector. If the card does not go fully in, do not force it and instead remove it and check for proper orientation or obstructions. Once fully inserted the Blue LED should go to solid ON while the Green LED should start blinking. Then push in the handle to latch the card into the carrier, the Blue LED should blink for a time and then go solid OFF while the Green LED goes solid ON.

To remove the card, pull out the hot-swap handle until it stops to unlatch the card from the carrier (but do not pull hard enough to remove the card itself yet). The Blue LED should blink for a time and then go solid ON. Once it does, pull the hot-swap handle straight out firmly to remove the card from the carrier.

2.6.2 Front Panel IPMI RS-232 Port

An IPMI RS-232 port is provided on the front panel for connecting to the MMC CPU. This port is used for configuration of AMC E-Keying using a menu interface (see subsequent section). A VadaTech cable (P/N CBL-DB9MUSB1) is available for converting this port into a DB9 serial port. The port setup is 115200-8-N-1-NOFLOW. The pin-out is as follows:

Pin	Signal
1	n.c.
2	RXD
3	TXD
4	n.c.
5	GND
SHIELD	CGND

Table 7: IPMI RS-232 port pin-out

WARNING: *This port uses the MicroUSB form factor but DOES NOT carry USB signaling. Therefore please be careful not to attach any USB device to the AMC520 board as damage could result.*

2.6.3 Front Panel FPGA RS-232 Port

A CPU/FPGA RS-232 port is provided on the front panel which is routed to the FPGA to enable it to be used by a soft CPU core in the FPGA if desired. A VadaTech cable (P/N CBL-DB9MUSB1) is available for converting this port into a DB9 serial port. The port setup is 115200-8-N-1-NOFLOW.

The pin-out is as follows:

Pin	Signal
1	n.c.
2	RXD
3	TXD
4	n.c.
5	GND
SHIELD	CGND

Table 8: FPGA RS-232 port pin-out

WARNING: *This port uses the MicroUSB form factor but DOES NOT carry USB signaling. Therefore please be careful not to attach any USB device to the AMC520 board as damage could result.*

2.6.4 Front Panel SFP+ Ports

The front panel hosts two SFP+ ports which route directly to FPGA SERDES ports. In addition to the SERDES the SFP+ ports provide RXLOS (RX Loss of Signal) and TX_FAULT (TX Fault) indications to the FPGA as well as TX_DISABLE (TX Disable) control from the FPGA.

2.6.5 Front Panel FPGA STATUS LEDs

There is a grouping of 16 STATUS LEDs on the front panel with the first 12 of them being green and the last four being yellow. These LEDs are connected to the FPGA for displaying arbitrary status information in the customer's application. The usage of these LEDs by the FPGA reference design is shown below (but does not constrain the customer's application to using them for these purposes):

LED	Reference Design Usage	LED	Reference Design Usage	LED	Reference Design Usage	LED	Reference Design Usage
0	ADC 0/1 Running	4	ADC 8/9 Running	8	MMCMs Locked	12	FPGA main reset
1	ADC 2/3 Running	5	DAC 0/1 Running	9	QDRII+ BIST OK	13	QDRII+ Calibrating
2	ADC 4/5 Running	6	SFP+ 0 SYNCed	10	SFP+ 0 Signal Detected	14	PCIe 4-7 PCIe TRN reset
3	ADC 6/7 Running	7	SFP+ 1 SYNCed	11	SFP+ 1 Signal Detected	15	PCIe 4-7 not x4

Table 9: STATUS LED usage in Reference Design

NOTE: The FPGA reference design implements a power-on lamp test mode for these LEDs. After the FPGA loads it will turn all of these LEDs on for one second then off for one second, after which it displays the status information as shown above. Also note that it is normal for the QDRII+ to not finish calibration if the board is ordered without QDRII+ chips.

2.6.6 Front Panel User LEDs

The front panel includes four green User LEDs which are controlled via the FPGA. The FPGA reference design uses these LEDs in the following manner (but does not constrain the customer's application to using them for these purposes):

LED	Reference Design Usage
U0	Backplane FPGA 1000Base-X Port 0 SYNCed
U1	Backplane FPGA 1000Base-X Port 1 SYNCed
U2	Backplane FPGA PCIe 4-7 Linked
U3	Backplane FPGA PCIe 8-11 Linked

Table 10: FPGA User LEDs

NOTE: The FPGA reference design implements a power-on lamp test mode for these LEDs. After the FPGA loads it will turn all of these LEDs on for one second then off for one second, after which it displays the status information as shown above.

2.6.7 Front Panel FPGA DONE LED

A green FPGA DONE LED is lit to indicate that the FPGA configuration loaded successfully.

2.6.8 Front Panel PCIe LEDs

The front panel includes eight green LEDs showing PCIe RX signal detection (Root Complex is sending PCIe signals) and eight green LEDs showing PCIe TX signal detection (FPGA is sending PCIe signals). These LEDs are driven by the on-board PCIe repeaters.

2.6.9 Front Panel FPGA JTAG Port

The front panel includes a JTAG port which is routed to the Xilinx Virtex-6 FPGA on the board when SW5[3] is set to OFF. This port can be connected to a Xilinx Platform USB II cable (or equivalent) and has the following pin-out:

Pin	Signal	Pin	Signal
1	GND	2	+2.5V
3	GND	4	TMS
5	GND	6	TCK
7	GND	8	TDO
9	GND	10	TDI
11	GND	12	n.c.
13	GND	14	n.c.

Table 11: FPGA JTAG Pin-out

2.6.10 Front Panel CLK IN Port

There is a CLK IN port provided on the front panel via an MMCX jack. This clock input is one possible source for the ADC clock distribution.

The clock input uses the following circuit:

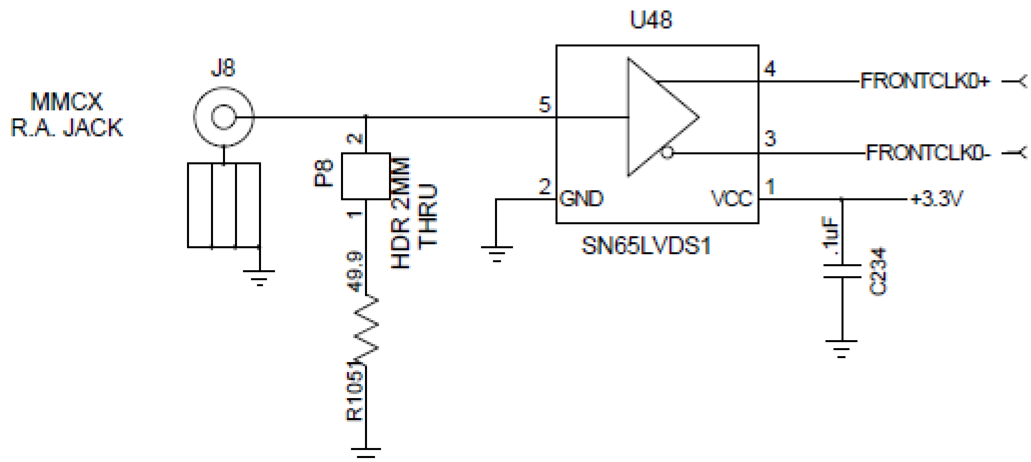


Figure 6: Front panel CLK IN circuit

The input is LVTTL w/ 5V tolerance.

2.6.11 Front Panel TRIG IN Port

The front panel includes a TRIG IN port via an MMCX jack which uses the following input circuit:

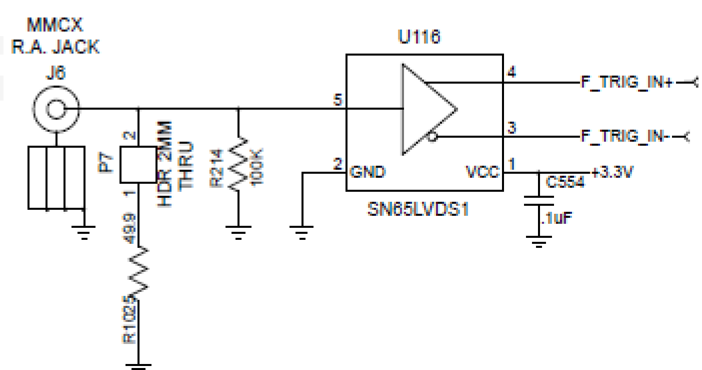


Figure 7: Front panel TRIG IN circuit

The input is LVTTL w/ 5V tolerance.

2.6.12 Front Panel TRIG OUT Port

The front panel includes a TRIG OUT port via an MMCX jack which uses the following output circuit:

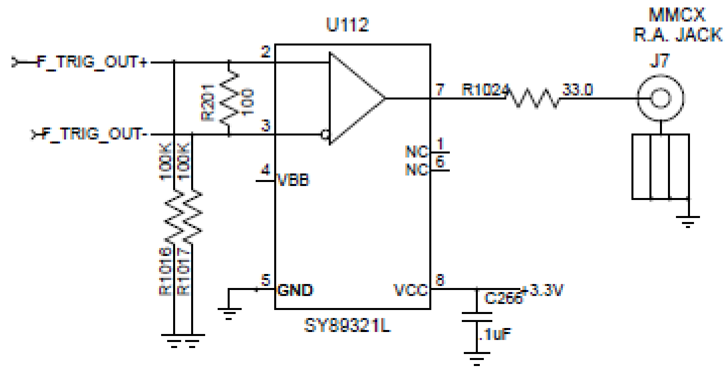


Figure 8: Front panel TRIG OUT circuit

The output provides a low voltage of between 0V and 0.5V and a high voltage of between 2.0V and 3.3V.

2.6.13 Front Panel User I/O

The front panel includes two Samtec TSS-105-04-G-D-RA 10-pin connectors which connect directly to the FPGA pins and include ESD protection circuits. These connectors are +2.5V compatible and provide a total of 16 I/O pins which may be used for any user input/output needs in the customer application. The directionality is determined by the FPGA image loaded. Please take care to follow the directionality of the FPGA image as there is no short-circuit protection. The pin-out of the connectors is as follows:

Pin	Signal	Pin	Signal
1	USERI/O-00	2	USERI/O-01
3	USERI/O-02	4	USERI/O-03
5	USERI/O-04	6	USERI/O-05
7	USERI/O-06	8	USERI/O-07
9	GND	10	GND

Table 12: USER IO 0/7 Connector

Pin	Signal	Pin	Signal
1	USERI/O-08	2	USERI/O-09
3	USERI/O-10	4	USERI/O-11
5	USERI/O-12	6	USERI/O-13
7	USERI/O-14	8	USERI/O-15
9	GND	10	GND

Table 13: USER IO 8/15 Connector

2.7 On-board ADC/DAC Clock Routing

The ADC clock distribution is handled by a set of external clock distribution buffers arranged in two stages. The resulting clock from the two stages is selected by the FPGA using the CLK_SEL[1:0]_[1:0] lines for stage 1 and CLK_SEL for stage 2 (refer to FPGA pin-out in VHDL Sources release package).

CLK_SEL	CLK_SEL[1:0]_1	CLK_SEL[1:0]_0	Meaning
'0'	Don't care	"00"	RTM Clock 0
'0'	Don't care	"01"	On-Board 125MHz
'0'	Don't care	"10"	(Reserved – clock off)
'0'	Don't care	"11"	(Reserved – clock off)
'1'	"00"	Don't care	Backplane TCLKA
'1'	"01"	Don't care	RTM Clock 1
'1'	"10"	Don't care	RTM Clock 2
'1'	"11"	Don't care	Front CLK IN

Table 14: Selecting the ADC clock source

The DAC chip is clocked from the FPGA. Therefore it is flexible as to what clock is used to drive it. The following diagram shows the ADC / DAC clock routing for the board.

NOTE: Higher resolution diagrams in PDF form can be found in the AMC520 VHDL Sources distribution package.

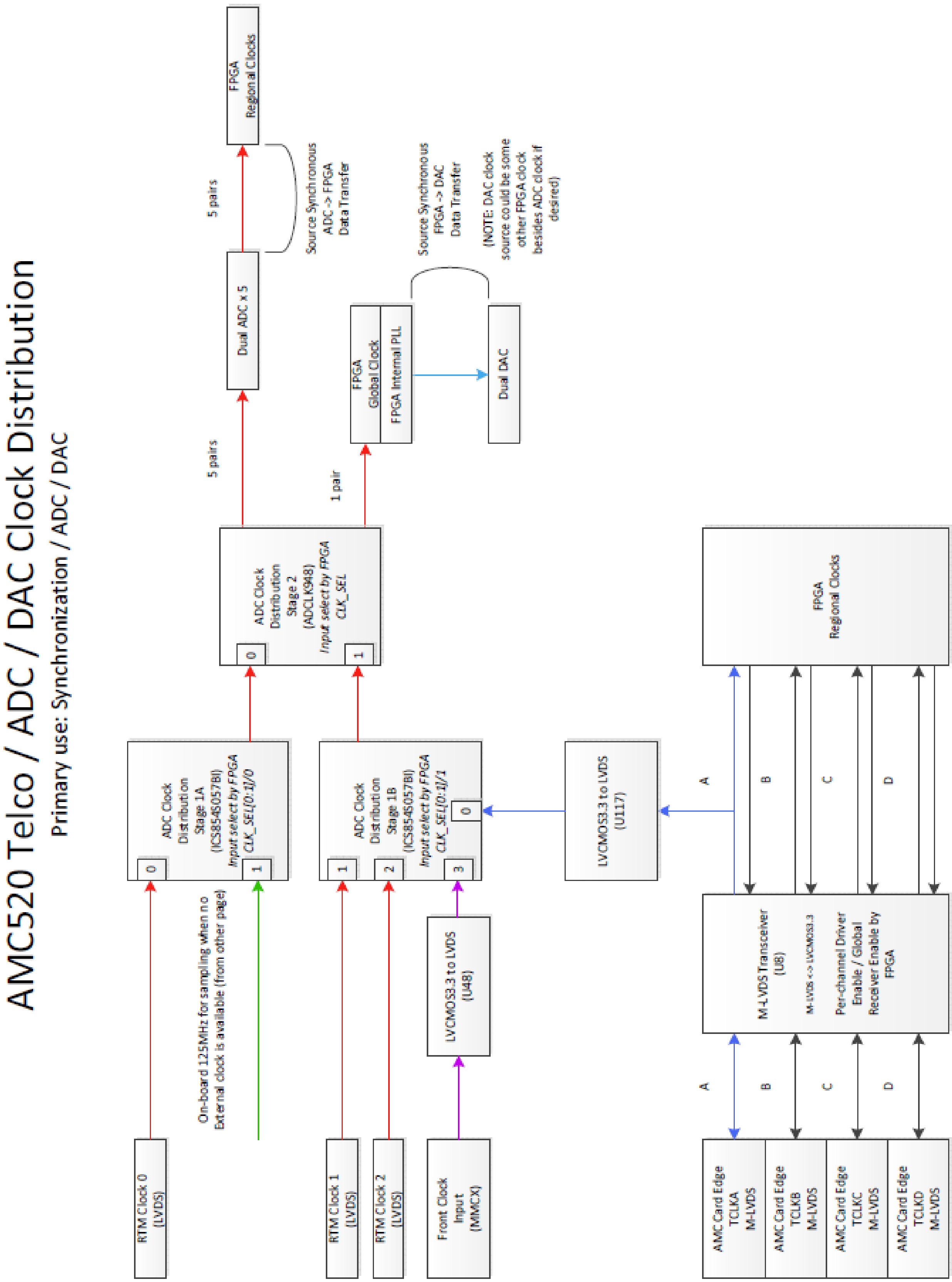


Figure 9: On-board ADC/DAC Clock Routing

2.8 On-board Trigger Routing

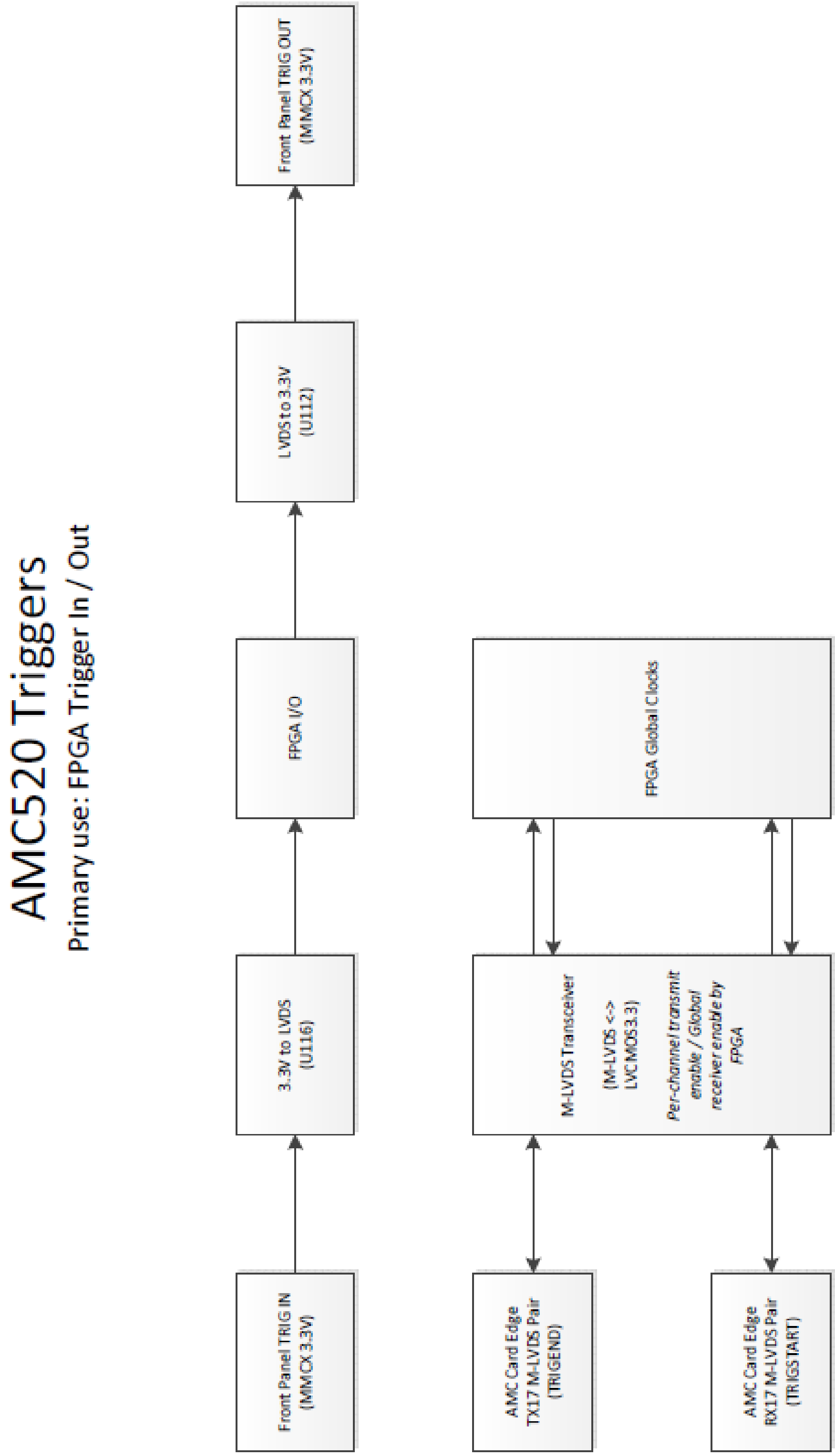


Figure 10: On-board trigger routing

2.9 On-board QDRII+ / IODELAY Calibration / Misc Clocking

Two programmable oscillators are provided for driving the QDRII+ and other IODELAY calibration / Misc clocking within the FPGA.

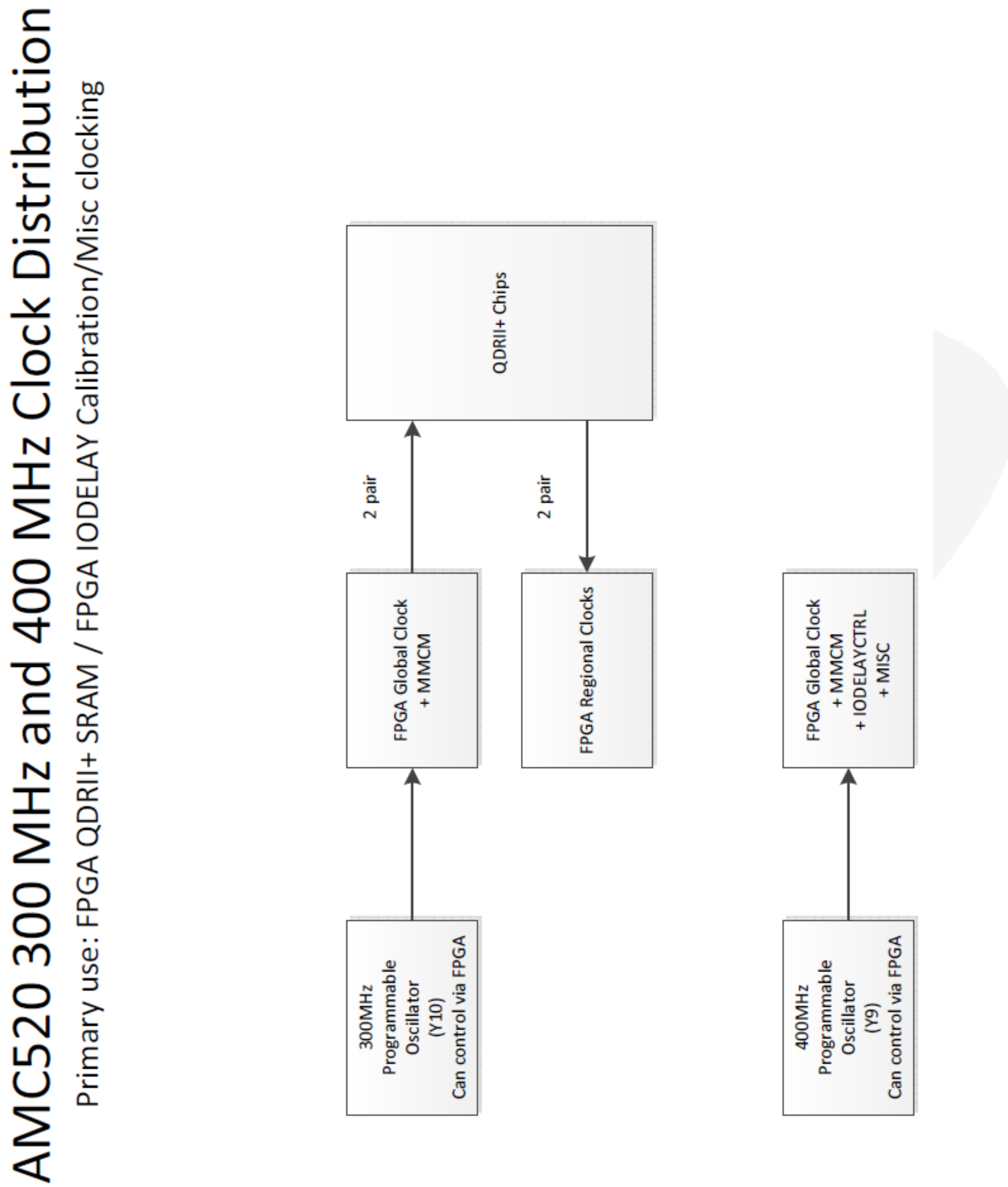


Figure 11: QDRII+, IODELAY calibration, and misc clock routing

2.10 Backplane/RTM Connections

Refer to the appendixes at the end of this document for backplane and RTM pin-outs.

2.1 FPGA Banking/Pinning/Floorplan

Detailed information on the FPGA Banking, Pinning, and Floorplan can be found in the AMC520 VHDL Sources package in the 'Docs' directory.

2.2 FPGA SERDES Backplane Interfaces

The board is designed to support flexible system interfacing to the backplane via the reprogrammable FPGA. Interfaces such as PCIe x1/x2/x4/x8 (Gen1 or Gen2), 1000Base-X, Aurora, and others are realizable. The FPGA reference design demonstrates 1000Base-X to AMC ports 0 & 1, PCIe x4 Gen 1 to AMC ports 4-7, and PCIe x4 Gen 1 to AMC Ports 8-11.

AMC Port	Reference Design
0	1000Base-X (hard core)
1	1000Base-X (hard core)
2	Connected but unused in reference design
3	Connected but unused in reference design
4	PCIe x4 Gen 1 Lane 0 (hard core)
5	PCIe x4 Gen 1 Lane 1 (hard core)
6	PCIe x4 Gen 1 Lane 2 (hard core)
7	PCIe x4 Gen 1 Lane 3 (hard core)
8	PCIe x4 Gen 1 Lane 0 (hard core)
9	PCIe x4 Gen 1 Lane 1 (hard core)
10	PCIe x4 Gen 1 Lane 2 (hard core)
11	PCIe x4 Gen 1 Lane 3 (hard core)
12	Connected but unused in reference design
13	Connected but unused in reference design
14	Connected but unused in reference design
15	Connected but unused in reference design

Table 15: FPGA reference design backplane SERDES interfaces

2.3 FPGA SERDES Front Panel Interfaces

The board is designed to support flexible system interfacing to the front panel SFP+ cages via the reprogrammable FPGA. Interfaces such as 1000Base-X, Aurora, and others are realizable. The FPGA reference design demonstrates 1000Base-X to SFP+ ports 0 & 1.

SFP+ Port	Reference Design
0	1000Base-X (hard core)
1	1000Base-X (hard core)

Table 16: FPGA reference design backplane SERDES interfaces

2.4 FPGA SERDES Reference Clocks

The AMC520 design attempts to provide the most flexible options for GTX clock forwarding to enable a wide variety of SERDES protocols while minimizing the number of clocks on the board to reduce noise to the ADCs.

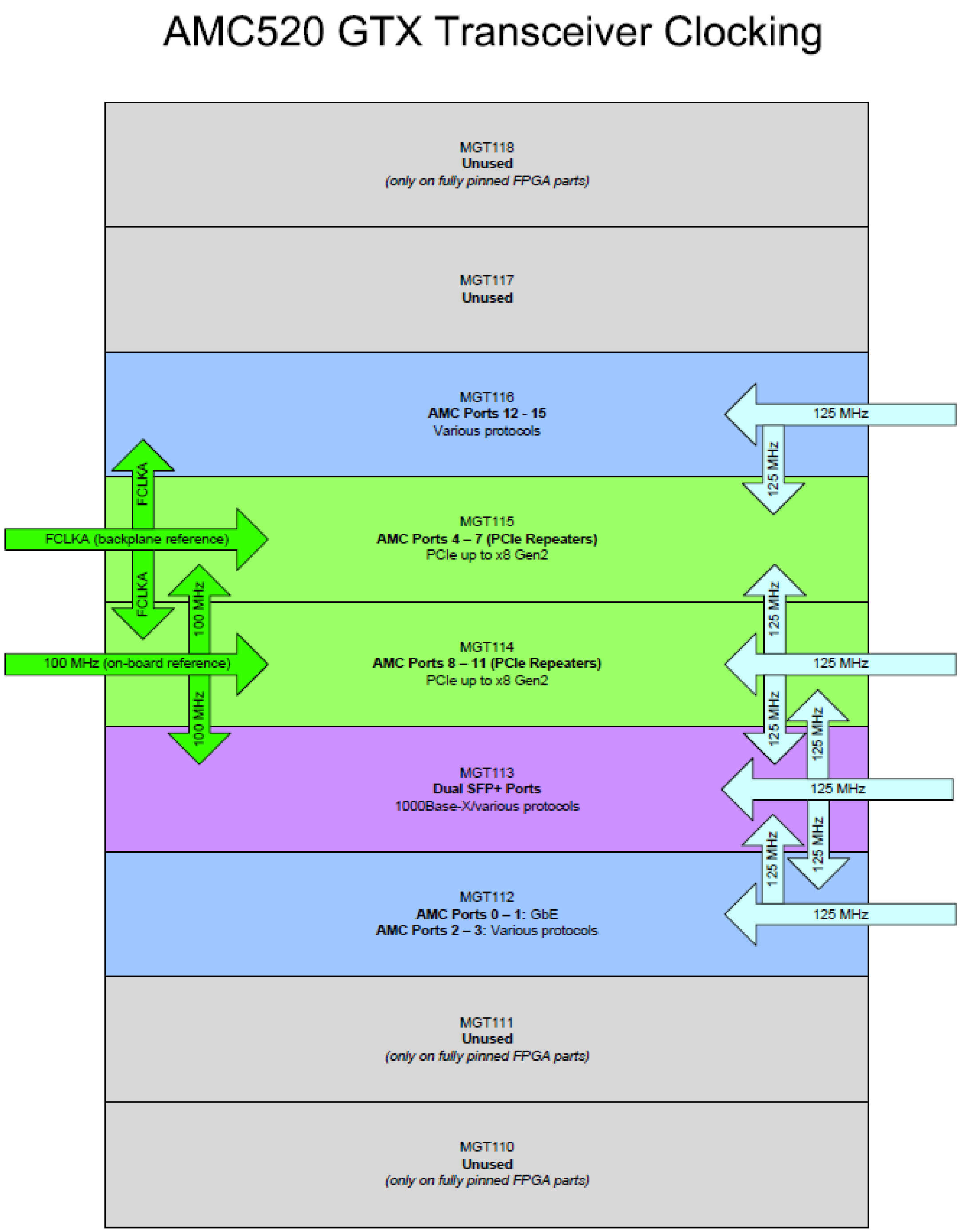


Figure 12: Virtex 6 GTX clock forwarding for AMC520

2.5 Backplane/On-board PCIe Clock Routing (CLK3/FCLKA)

There are two different ways to clock the PCIe cores in the FPGA. The preferred way is to use the backplane CLK3/FCLKA which is provided into the MGT115 bank, by using the backplane clock coming from the MCH then the PCIe can work with both spread-spectrum and non-spread-spectrum clocking. An on-board 100MHz oscillator is also provided into the MGT114 bank which enables non-spread-spectrum clocking only if a backplane PCIe clock is not available. Either clock can be used by either or both PCIe banks via internal clock forwarding in the Virtex-6.

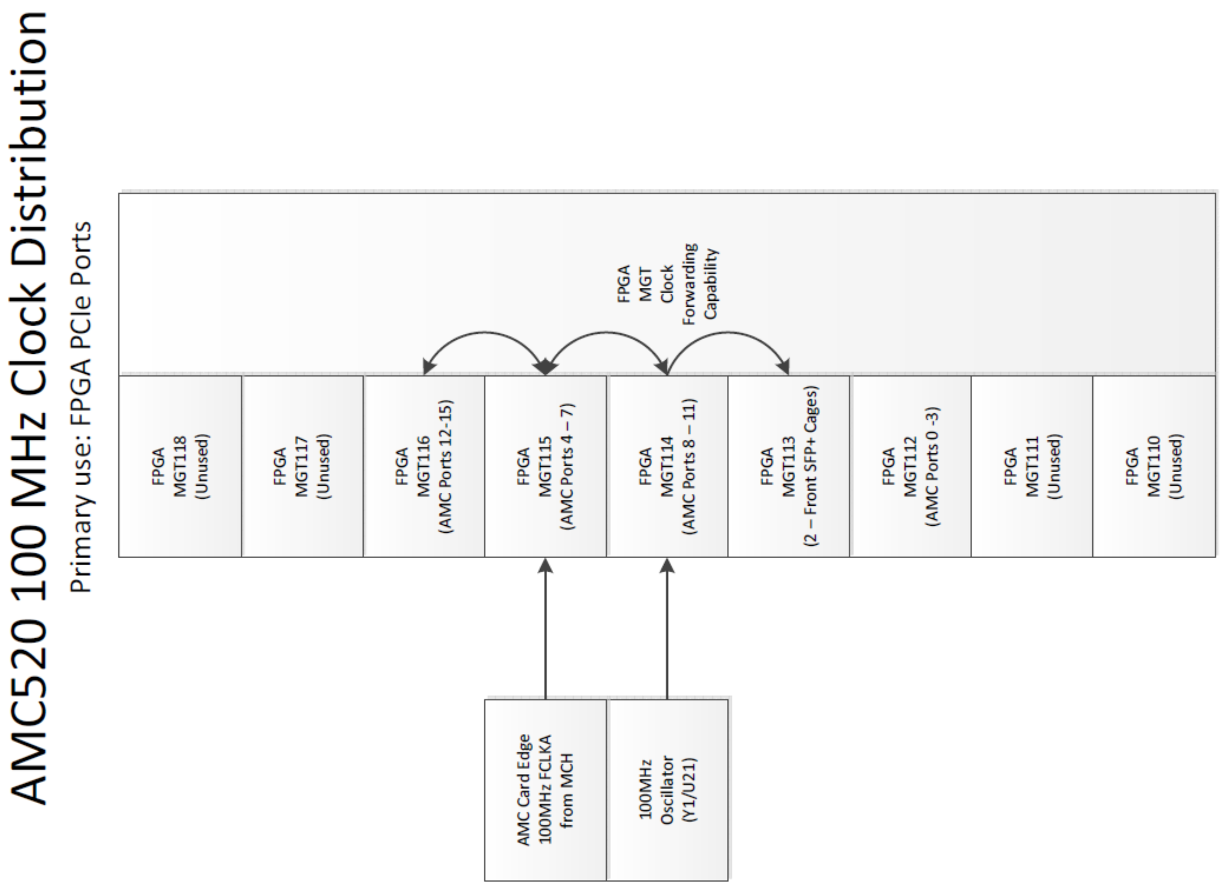


Figure 13: PCIe clock routing

NOTE: The FCLKA is used ONLY for PCIe clocking on the AMC520 in keeping with the latest AMC specifications. It cannot carry an arbitrary CLK3 signal from the backplane through to the FPGA. Please consult the VadaTech Telco/GPS Clock Configuration Guide for further details.

2.6 On-board 125MHz Clock Routing

An on-board 125MHz clock is provided for use in various SERDES protocols (i.e. 1000Base-X) and also as a possible source for ADC clocking.

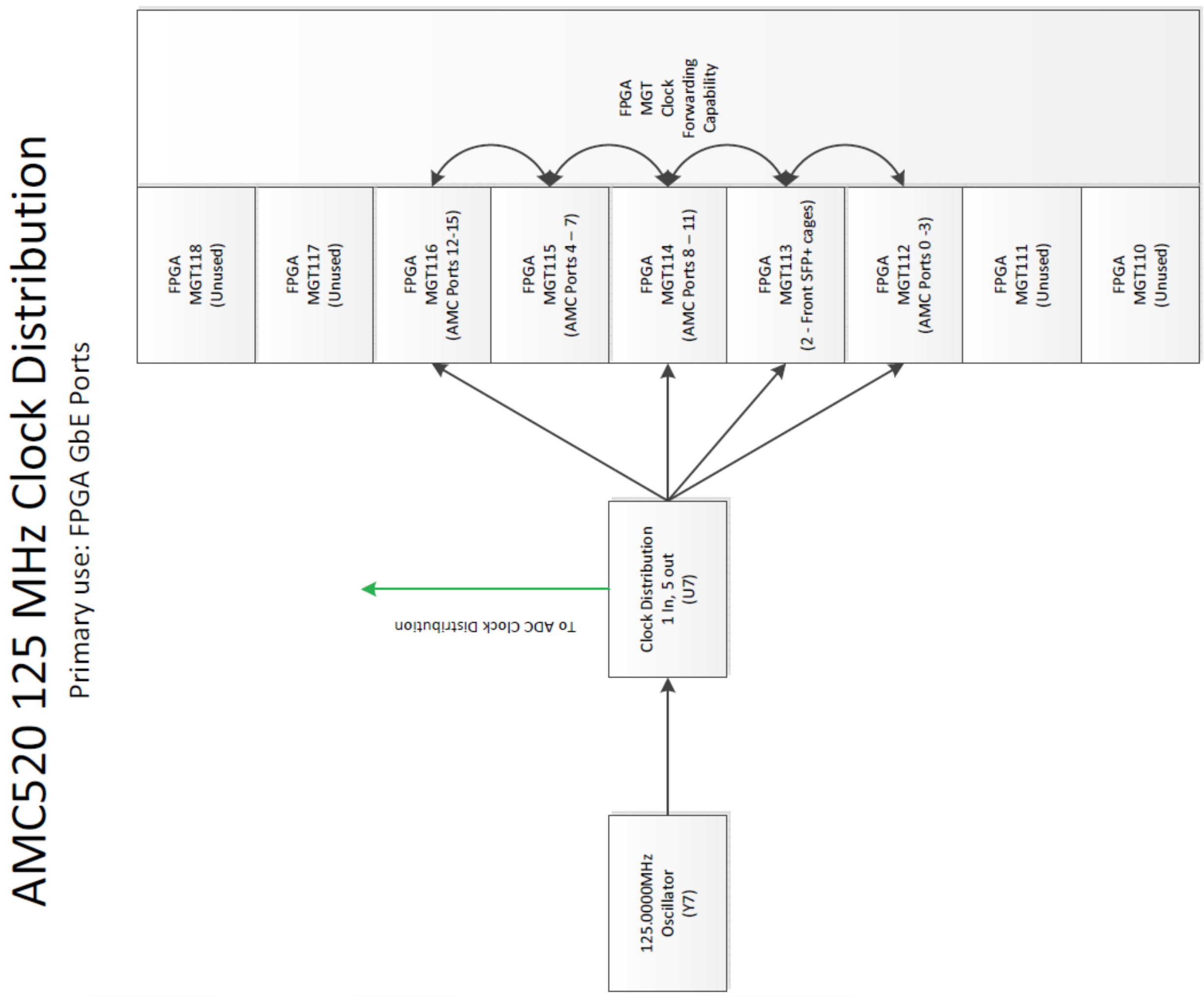


Figure 14: 125MHz clock routing

2.7 ADC Channel Coupling (Magnetic vs. OpAmp)

The 10-channel ADC channel coupling configuration can be ordered standard from the factory as all Magnetic (F=0) or all OpAmp (F=1). However, some customers may find it necessary to modify this coupling after purchase to either completely change from one type to the other or to change only some channels from one type to the other.

WARNING: *This section describes physical board modifications to be carried out at the customer site. These modifications will void your warrantee if not performed by a skilled electronics rework technician. The modifications are very simple since they only entail moving some zero-ohm resistors, however, VadaTech will not be responsible for customer-caused damage to the board.*

The general location of the zero-ohm resistors is on the top (component) side of the board as highlighted by the orange rectangles below:

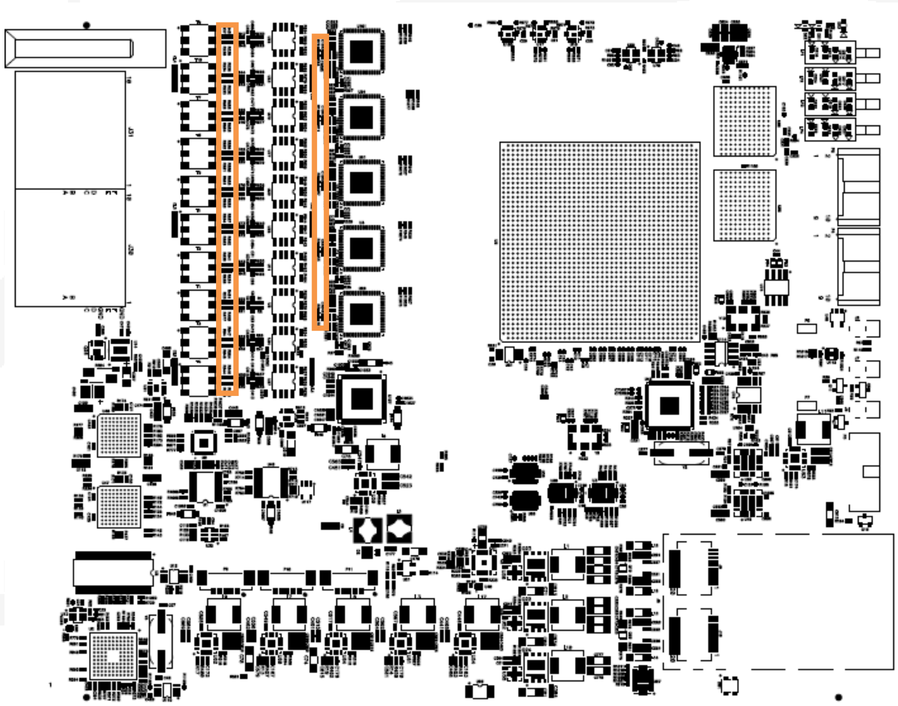


Figure 15: ADC Coupling Zero-Ohm Resistors Location

Each channel is routed either to a magnetic or an op-amp circuit. This routing is done by way of three-pad 0402 size zero-ohm resistors on the board. The center pad connects to the ADC chip-side circuit and the other two outer pads connect to the magnetic or op-amp circuits leading back to the RTM. Therefore, changing the coupling for the ADC chip is simply a matter of shifting the zero-ohm resistors over on the pads using proper rework techniques.

There are connections for both the differential analog signals and the common mode. Both must be matched for a proper configuration. Also, a single common mode signal comes from each dual-channel ADC chip and is shared between two adjacent channels. Therefore the coupling configuration must be changed in pairs and not individually.

The following table lists the resistors which must be mounted for each coupling configuration of each channel/pair of the AMC520 Rev C board:

ADC Channel	Channel Pair	Magnetic	OpAmp
0	A	R937/R936/R44	R940/R935/R78
1		R934/R933/R44	R939/R932/R78
2	B	R685/R684/R11	R688/R683/R12
3		R682/R681/R11	R686/R680/R12
4	C	R661/R660/R7	R665/R659/R8
5		R664/R663/R7	R667/R662/R8
6	D	R639/R638/R3	R641/R634/R4
7		R33/R477/R3	R491/R490/R4
8	E	R94/R645/R5	R647/R584/R6
9		R644/R640/R5	R646/R635/R6

Table 17: ADC Channel Coupling Resistors

Two example diagrams are shown below. The first diagram shows where the three-pad resistors should be mounted to make channels 0 and 1 use Magnetic coupling. The second diagram shows where the three-pad resistors should be mounted to use OpAmp coupling. The other pairs of channels follow a very similar layout to these example channels.

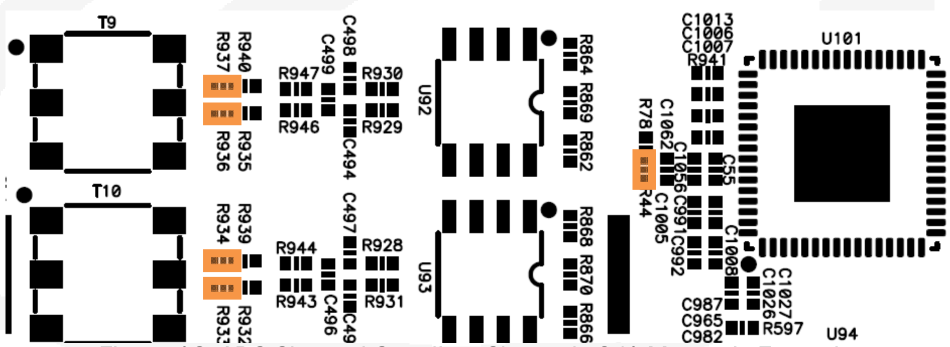


Figure 16: ADC Channel Coupling: Channels 0/1 Magnetic Example

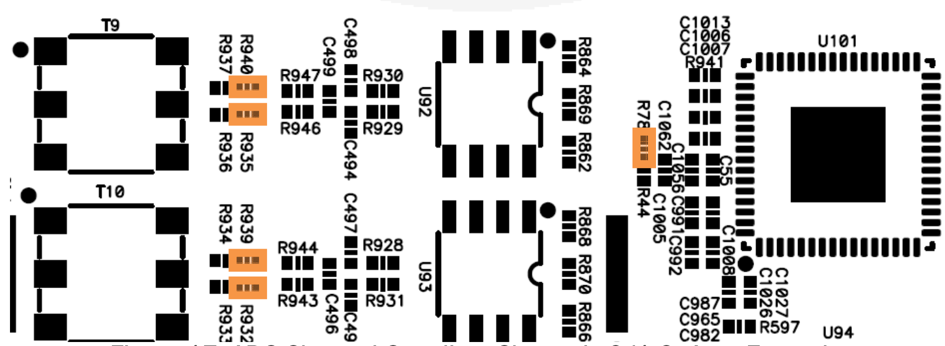


Figure 17: ADC Channel Coupling: Channels 0/1 OpAmp Example

3 IPMI MMC Serial Menu Interface

The IPMI MMC microcontroller handles communication with the AMC carrier and must provide E-Keying information to describe the SERDES ports used on the AMC card-edge connector. The MMC configuration for E-Keying initially ships with a matching set of E-Keying records for the FPGA reference design, meaning two GbE ports on AMC Ports 0 & 1, PCIe on AMC ports 4-7, and XAUI ports on AMC Ports 8-11. If you change the FPGA design to use different ports or protocols then you should use the MMC controller's menu system to configure the appropriate e-keying records. Using incorrect E-Keying records can result in unexpected behavior.

The interface to the MMC is via the IPMI RS-232 port (refer to the hardware overview section for more information). The port settings are 115200-8-N-1-NOFLOW. The configuration is described in the following sections.

3.1 E-Keying Configuration

Electronic Keying or E-Keying is a complex subject that is beyond the scope of this document. The rest of this section assumes familiarity with the relevant sections of PICMG® AMC.0 R2.0 Advanced Mezzanine Card Base Specification and the AMC.1, AMC.2 and AMC.3 subsidiary specifications.

The FPGA on the AMC can be programmed to support multiple link types in the Common Options region and the Fat Pipes Region of the connector. In order for the AMC to function properly in a MicroTCA or ATCA carrier, the AMC's FRU Information must correctly describe the link types provided by the FPGA. To set the E-Keying information, type **`ekey`** and press Enter. This will display a list of common link descriptors. Any link descriptor can be enabled or disabled by typing the corresponding number and pressing Enter. When the configuration is correct, type "**`save`**" and press Enter to save it in the AMC's FRU Information. To cancel your changes and return to the main menu, type "**`cancel`**" and press Enter.

4 IPMI Sensors

The AMC520 Management Controller monitors the following sensors:

Sensor Number	Name	Description
0x90	VT AMC520 HS	AMC.0 Hot-Swap Sensor
0x10	VT AMC520 T1	Intake Air Temperature
0x11	VT AMC520 T2	Exhaust Air Temperature
0x18	VT AMC520 Tint	Board Temperature near FPGA
0x19	VT AMC520 Text	FPGA Die Temperature
0x20	VT AMC520 12V	12V Input Power
0x28	1.0V	1 Volt Rail Voltage
0x29	1.5V	1.5 Volt Rail Voltage

Table 18: MMC Sensors

To access the sensors please refer to your AMC carrier's documentation.

5 FPGA Reference Design

The FPGA is fully customizable and it is expected that the customer will need to provide their own custom FPGA design to enable application-specific data processing within the fabric. However, a reference design is provided to demonstrate the basic board hardware functionality and to act as a manufacturing/ acceptance test.

5.1 FPGA Reference Design External Interfaces

The reference design demonstrates the following ports:

- 2 – 1000Base-X Ethernet to AMC Ports 0 & 1 (link only, no traffic)
- 2 – 1000Base-X Ethernet to SFP+ Ports 0 & 1 (link only, no traffic)
- PCIe x4 Gen 1 to AMC Ports 4-7 (full register interface)
- PCIe x4 Gen 1 to AMC Ports 8-11 (configuration space only)
- 10-channels (5 chips) of ADC w/ BIST verifier and data path and synchronizer
- 2-channels (1 chip) of DAC w/ built-in signal generation
- QDRII+ 72-bit SRAM controller
- 4 - User LEDs w/ diagnostic display
- 16 - Status LEDs w/ diagnostic display
- Clock/trigger routing interfaces
- SYSMON temperature/voltage readings
- FPGA RS-232 port (loopback)
- 16 - User I/O Outputs
- MRT520 DensiShield external loopback test for RTM data/clock interfaces

NOTE: These are the interfaces chosen for the reference design, but the FPGA's pin-out is designed to enable a great deal of flexibility for customer-specific applications. Many different styles of interfaces are possible to both the AMC backplane and front panel through custom FPGA development at the customer's site.

5.2 FPGA Reference Design Internals

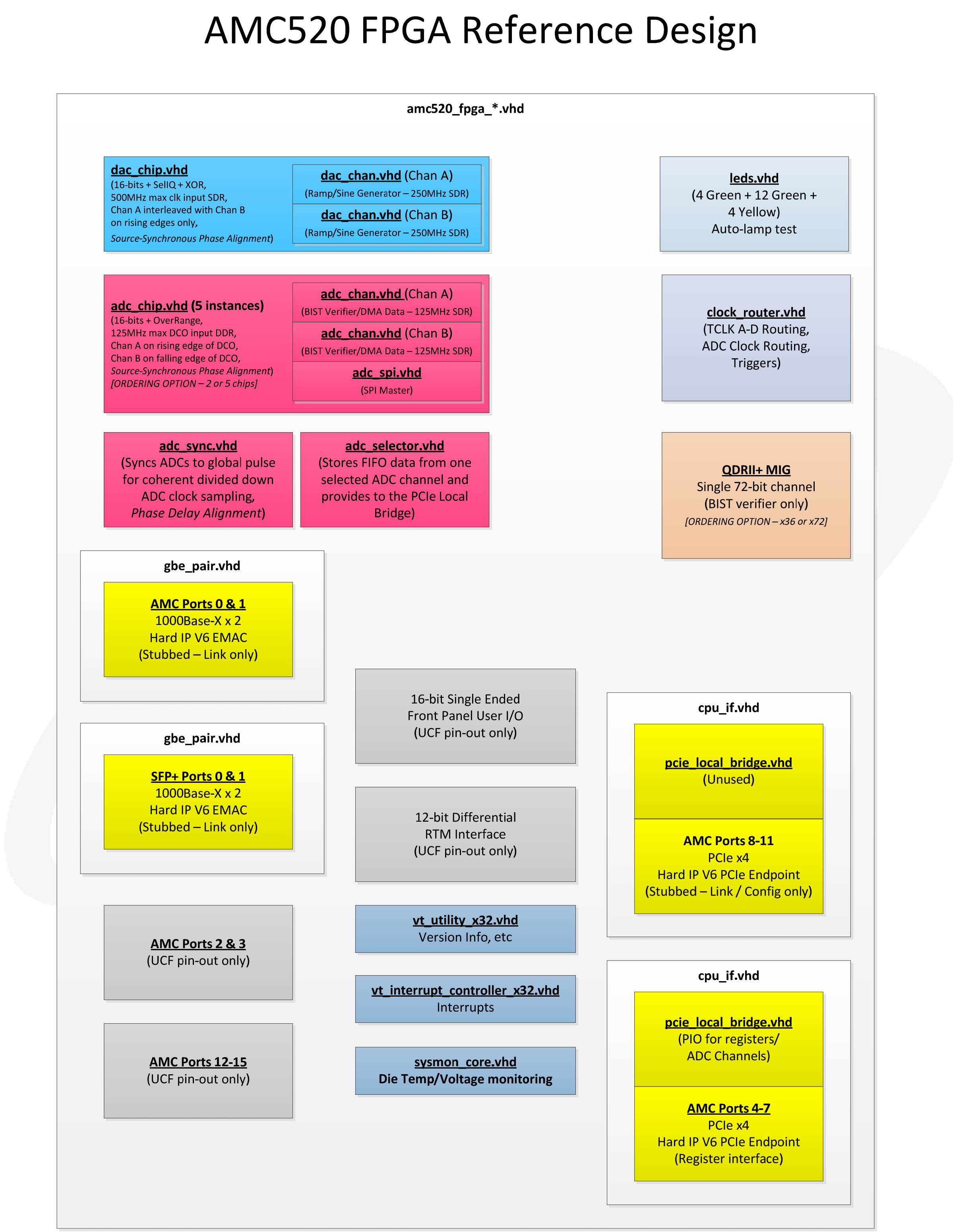
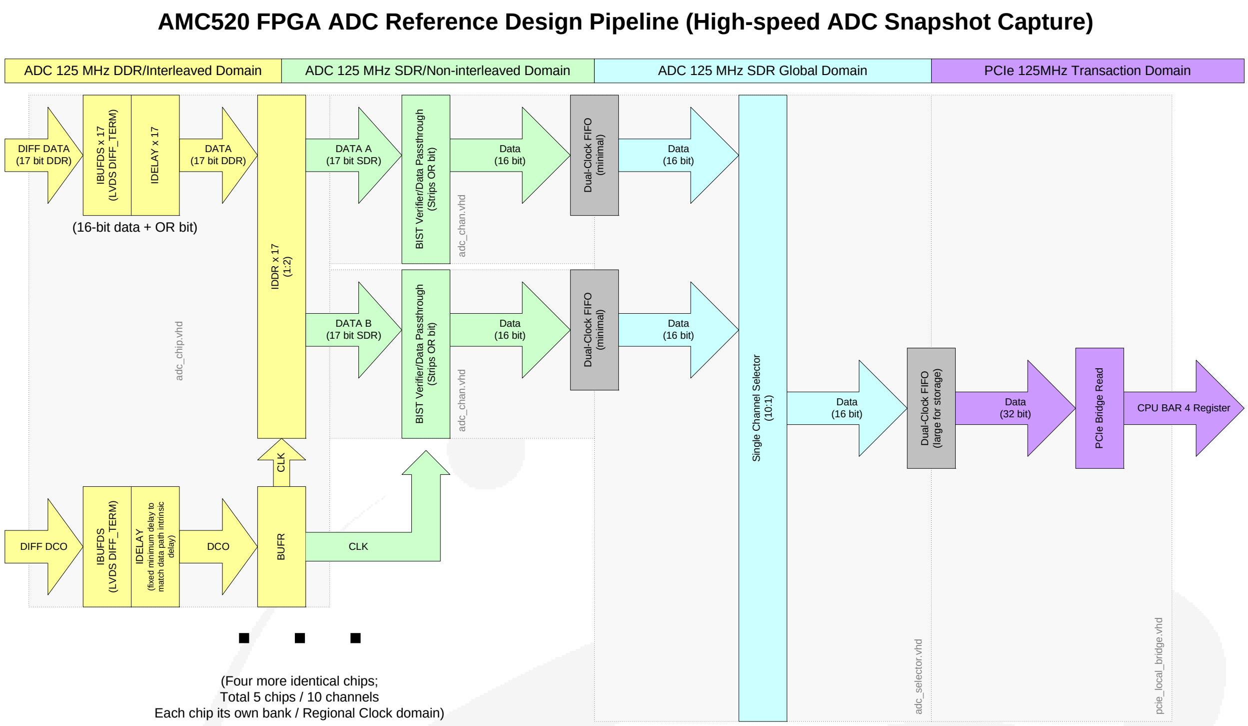


Figure 18: AMC520 FPGA Top-Level Reference Design Diagram



SNAPSHOT STORAGE AVAILABLE:

~128 Kilo-samples

DISCLAIMER: It is assumed that an end-user design is likely to perform DSP to intelligently reduce the data and/or streaming operations to extend the sampling time (typically to support continuous sampling), but that is beyond the scope of this example design. The example design captures a snapshot in one pass and then the snapshot is read by the CPU in another pass for the purpose of creating a raw data file for analysis. The limitations of the example design are not intended to be nor should be considered as limitations to the end-user design.

NOTE: Transfer of a single ADC channel at a time is supported by the reference design for sake of simplicity.

Figure 19: AMC520 FPGA ADC Reference Design Diagram

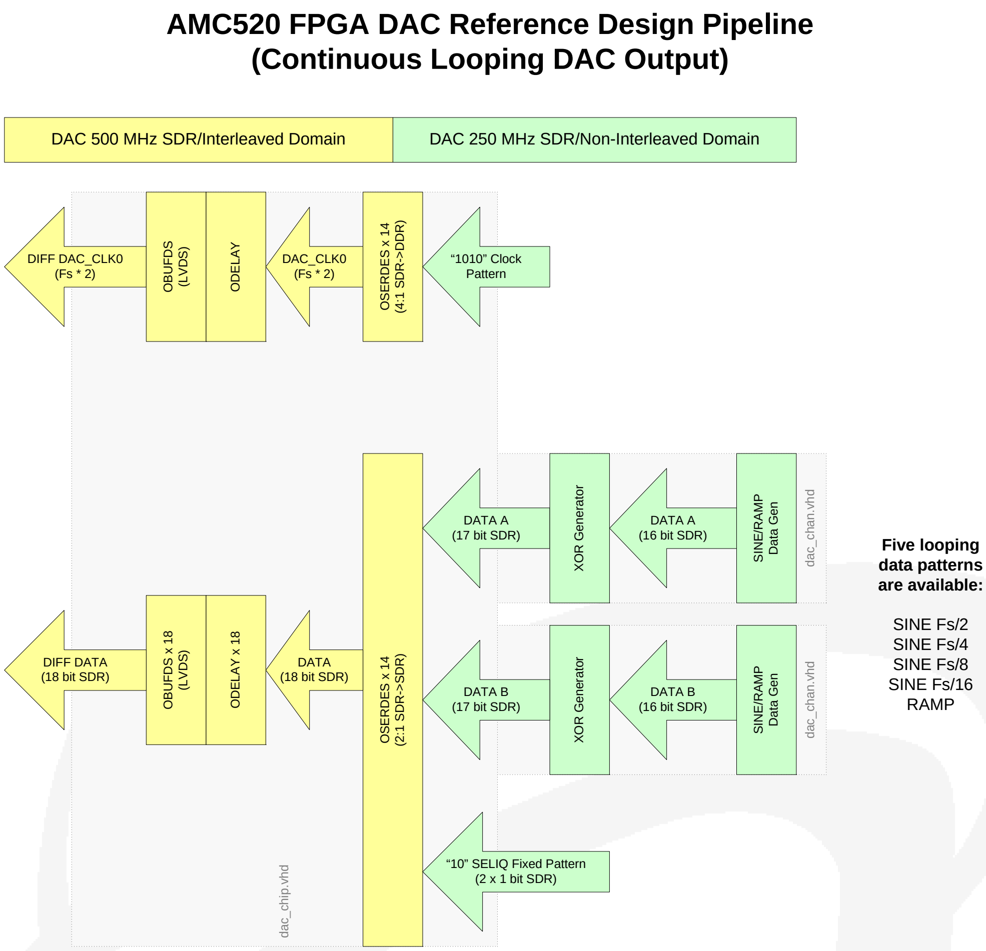


Figure 20: AMC520 FPGA DAC Reference Design Diagram

NOTE: Full-size versions of these diagrams are available in the AMC520 VHDL Sources package under the 'Docs' directory.

In addition to the Xilinx SERDES/MIG cores which make up some of the external interfaces in the reference design, there are various other VadaTech cores in the design which are described below.

5.2.1 PCIe Bridge (pcie_local_bridge.vhd)

The PCIe Bridge implements a state machine which converts PCIe Transaction layer packets into internal register bus transactions. It supports up to 32-bit PIO reads and up to 32-bit PIO writes. Byte enables are supported for register accesses so individual bytes may be read/written if desired.

The PCIe bridge also includes a state machine for handling in-band interrupt signaling to the host processor.

5.2.2 Interrupt Controller (vt_interrupt_controller_x32.vhd)

The interrupt controller consolidates the level-sensitive interrupt lines coming from several internal and external sources to provide one single master interrupt line to the PCIe Bridge. The interrupt controller also supports a Bit Change Interrupts which are effectively edge detectors combined with an event capture mechanism used for alerting the software any time monitored lines such as the thermal alert line change state.

5.2.3 Clock Router (clock_router.vhd)

The Clock Router enables flexible routing of off-board and on-board clocks. It also enables/disables various input/output buffers. See the register specification that follows for details of routing sources and targets.

5.2.4 LED Controller (leds.vhd)

The LED Controller allows various statuses to be reflected onto the front panel User LEDs. Refer to the Hardware Overview section for status information displayed.

5.2.5 ADC Chip Controller (adc_chip.vhd)

This core controls one instance of a dual-channel ADC chip and includes instances of the two ADC channels which the chip handles in addition to an instance of a SPI Master controller to facilitate communication to ADC chip registers. This core includes the basic source-synchronous clocking and I/O interfacing for the ADC chip interface to the FPGA. It includes the ability to delay the data relative to the clock if needed.

5.2.6 ADC Channel (adc_chan.vhd)

This core implements a single de-interleaved ADC channel with a BIST verify and data channel which connects to the ADC Selector.

5.2.7 ADC SPI Master (adc_spi.vhd)

This core implements a SPI Master for communicating with the ADC chip registers.

5.2.8 ADC Synchronizer (adc_sync.vhd)

This core outputs synchronization pulses to the five ADC chips on the board so that when they are in a divided down clock mode they will all sample coherently on the same clock edge. This core includes the ability to delay the sync pulses relative to the ADC global clock if necessary. NOTE: Although the FPGA provides this example of how to generate sync pulses, the reference design doesn't use the ADC chips in a divided down mode so the effect of the sync pulse generation is only visible by probing the signals on the board or if the customer sets up their own divided down mode for the ADC chips.

5.2.9 ADC Selector (adc_selector.vhd)

This core picks off the data from one of the ten ADC channels and stores it into one large storage FIFO. This storage FIFO can then be read out by the external CPU via PCIe to collect ADC sample data into a file.

5.2.10 DAC Chip Controller (dac_chip.vhd)

This core controls the dual-channel DAC chip and includes two instances of the DAC channels the chip handles. This core includes the basic source-synchronous clocking and I/O interfacing for the DAC chip interface to the FPGA. It includes the ability to delay the data relative to the clock if needed.

5.2.11 DAC Channel (dac_channel.vhd)

This core implements a DAC channel with data generation for nine different looping data patterns: All zeros, All ones, All mid-level, Ramp, SINE Fs/2, SINE Fs/4, SINE Fs/8, and SINE Fs/16, or a user supplied level. The sine patterns generate continuous sine waves but only utilize a limited number of DAC symbols. The ramp pattern generates every DAC symbol from 0x0000 through 0xFFFF and then wraps around and repeats (creating a saw-tooth wave).

In addition to the data generation capabilities, the DAC channel cores can also loop-through the ADC data from any of the 10 ADC channels such that the analog output will be a reproduction of the analog input.

5.2.12 BPI Flash Core (bpi_flash.vhd)

A BPI Flash core is present which enables access to the BPI Flash on the board via PCIe. This core enables the software running on the host CPU to reprogram the FPGA configuration file and trigger re-configuration. The software has a feature which enables this

to be seamless by saving off and then restoring the PCIe core configuration info across the reconfiguration of the FPGA.

5.2.13 SYSMON Core (sysmon_core.vhd)

A SYSMON core is present which captures the internal die temperature, internal voltage, and auxiliary voltage.

5.2.14 Utility Core (vt_utility_x32.vhd)

A utility cores is present which contains read-only identification information such as version, signature, etc. It also contains some read-only registers which reflect current status information that can change over time. There is also a scratch register present in this core which is useful for bus testing.

5.2.15 Miscellaneous Helper Cores

Some general-purpose VadaTech cores, which were not described previously, are also included in the reference design as follows:

- General Purpose Arbitrary Clock Enable Divider (vt_clocken_div_arbitrary.vhd)
    - Provides a re-usable way to slow down portions of the design while still using the same clock
- General Purpose Reset Synchronizer (vt_reset_sync.vhd)
    - Provides a re-usable asynchronous assertion/synchronous de-assertion reset
- General Purpose Clock Synchronizer (vt_multi_sync.vhd)
    - Provides a re-usable clock domain crossing for individual signals to help mitigate meta-stability

5.2.16 Xilinx IP Cores

The following cores from Xilinx are used:

- Virtex-6 Integrated Block for PCI Express (x2)
    - Wraps the two Virtex-6 embedded PCIe cores
    - Provides the basic control/status/data interface for the reference design
    - Provides for basic backplane verification for PCIe

- Virtex-6 Embedded Tri-mode Ethernet MAC Wrapper (x2)
  - Wraps the four Virtex-6 embedded MAC cores and GTX transceivers
  - Used in 1000Base-X mode in this design for backplane/SFP+ verification

- Memory Interface Generator
  - Used to generate a QDRII+ 72-bit memory controller
  - Includes built-in BIST engine

- FIFO Interface Generator
  - Used for various clock domain crossing and data storage purposes

6 Customer FPGA Development

The reference FPGA design combines IP cores from Xilinx with VadaTech custom VHDL code. This design can be changed or replaced by the customer to allow for custom DSP/control solutions that are tailor-made to take full advantage of the Xilinx Virtex-6 FPGA.

It is expected that if the customer wishes to synthesize a new FPGA image, that they will have access to both Xilinx ISE v13.4 (full version required to support the Virtex-6 chip on the board) as well as the necessary IP cores. Xilinx may require per-core licenses even if they are free. The Xilinx ISE tool license and core licenses are NOT included as part of the purchase of the AMC520 card.

LEGAL NOTICE: The VadaTech custom VHDL code included in this reference design is the intellectual property of VadaTech Incorporated. Permission is granted to use the VadaTech custom VHDL code royalty-free in customer designs targeting the VadaTech AMC520 card only. Redistribution to third parties or use of this code for any other purpose is strictly prohibited. VadaTech is not responsible for damage or loss caused by reprogramming of the FPGA by the customer. Use caution when changing the reference design or creating your own design as it is possible to damage components on the AMC520 board or other attached boards/equipment.

If a completely new design is desired it is recommended that at a minimum the UCF files from the reference design be used since they provide the complete pin-out of the FPGA on the AMC520 board. The provided source files were used to create the flash image that is shipped on the board from VadaTech.

6.1 Modular FPGA Project Design

The FPGA reference design uses a modular and flexible design for supporting multiple densities, speed grades, and control interfaces. Each unique combination of these variables is wrapped into an ISE project for the combinations that are currently supported (contact VadaTech if your desired combination is not listed). The following table shows how the projects use the available high-level modules to create unique supported product variations:

High-Level Module	AMC520-XXX-42X-1XX
amc520_fpga_low_density.ucf	X
amc520_fpga_amc_pcie_2x4.vhd	X

Table 19: Modular FPGA Project Variations

When reviewing the VHDL sources it is important to understand the thinking behind the project layout. In the root of the project folder you will find the VadaTech-developed VHDL source files. In this directory you will also find an **`amc520_fpga_coregen.xise`** project file which is used to generate all of the Xilinx IP cores into the **`ipcore_dir`** directory. The contents of this directory include only the Xilinx generated files without modification. The coregen project is not used to create an actual FPGA image, it is only used to create common Xilinx IP cores for use by the targeted sub-projects.

Sometimes it is necessary for VadaTech to modify the Xilinx generated files for special situations such as re-arranging clocking resources or to flip polarity of a SERDES pair, etc. When these situations arise we create a new directory with a prefix of **`Customized_`** (such as **`Customized_1000BaseX`**) where we will copy ONLY the Xilinx generated files that need direct modification and we make the changes in this location. This ensures that the customizations are not lost if the cores are regenerated. However, if/when the cores are regenerated and the parameters to the core are changed, you will need to perform a three-way merge between the newly generated IP files in the sub-directories of **`ipcore_dir`** and the **`Customized_*`** directories. This three-way merge needs to ensure that the newly generated options are incorporated while simultaneously preserving the VadaTech customizations. When regenerating IP with different parameters please also ensure that the changes are updated to the **`amc520_lib.vhd`** file too since various generics are listed in this file and will take precedence over the files in **`ipcore_dir`**.

Finally, a sub-directory containing the ordering options of the board is present (such as **`AMC520_XXX_42X_1XX`**) which includes the actual project file used to generate the FPGA images for the named board configuration by combining the appropriate source files in the root, **`Customized_*`** sub-directories, and **`ipcore_dir`** sub-directories.

6.2 FPGA Development/Debug Cycle

During FPGA development various different programming mechanisms are supported:

1) Configure FPGA directly via front panel or backplane JTAG using Xilinx Impact or equivalent and a .bit file. Remember to set the JTAG front/back switch setting on the board. The configuration remains only during the current power-up. No PCIe activity should be taking place to the AMC520 while the FPGA is reconfigured or the host CPU could lock-up, crash, etc. If this configuration is done after the host CPU's operating system has configured the PCIe BARs then it will be necessary to reboot the host CPU before attempting to use the PCIe bus to the AMC520. This will cause the host CPU to rescan the PCIe bus and configure the BARs appropriately.

2) Program FPGA's BPI Flash via front panel or backplane JTAG using Xilinx Impact or equivalent using a .mcs file. Remember to set the JTAG front/back switch setting on the board. Then reset/power-cycle the board to have the FPGA configure itself from the BPI flash. The configuration remains permanent and will take effect at every power-up. No PCIe activity should be taking place to the AMC520 while the FPGA's BPI flash is reprogrammed or the host CPU could lock-up, crash, etc. While programming the BPI flash the Xilinx Impact tool downloads its own FPGA core in order to facilitate programming and during this time the PCIe link to the AMC520 will go down.

3) Program the FPGA's BPI Flash via PCIe using the amc520tool and amc520_fpga.ko driver provided using a .bin file. During this process the PCIe configuration data that the operating system originally setup will be preserved, then the BPI flash programmed, then the FPGA will be instructed to reconfigure itself from the BPI flash, and finally the PCIe configuration data will be restored into the FPGA's PCIe core. The new FPGA image can then seamlessly be used without the need to reboot the host CPU. The configuration remains permanent and will take effect at every power-up.

NOTE: Approach #3 is only valid if the PCIe core's configuration space has not been modified in the new FPGA image compared to the one that is running at the time of the re-programming. If PCIe BARs are being added or resized, etc the safest thing to do is use approach #2 or alternatively the customer could create a variation of this scheme in the software that simply programs the BPI flash and then reboots the host CPU without attempting to restore the previous state. Attempting to restore the previous PCIe configuration state into an FPGA PCIe core that has been re-structured may result in the host CPU locking up, crashing, etc.

The .bin and .bit files can be created via Xilinx ISE; the reference design project is setup to create both.

The .mcs file can be created by encapsulating a .bit file using Xilinx Impact; the reference design project is setup to do this as well.

7 Host-side Software Support

The AMC520 board includes access to a software tool and device driver which are used to control the FPGA reference design from an external PCIe Host CPU (not included) such as an x86 PrAMC or desktop PC w/ VadaTech PCI101 adapter. The software support includes the following:

- amc520_fpga.ko: Device driver module to control the FPGA via PCIe
- amc520tool: Tool for controlling the device driver/FPGA

Sources are made available for the device driver and tool so that they can be used as an example for your own application-specific designs.

NOTE: Due to the complete re-programmability of the FPGA, it is not possible to make a universal AMC520 device driver/tool application. The driver/tool provided matches the FPGA image provided as the reference design. If the FPGA image is replaced or changed by the customer then it follows that the customer will also need to update the device driver/tool appropriately.

7.1 AMC520 Device Driver

To compile the AMC520 device driver:

1) Enter the **`driver`** directory
2) Modify the **`Makefile`** if necessary to point to your Linux kernel source directory
3) Type **`make`**

The device driver is loaded by issuing the following command:

```
insmod /modules/amc520_fpga.ko
```

The device driver supports open, close, poll/select, and ioctl operations from the application. Generally the application should **`open()`** the device, then make an ioctl call to get information about the device if desired.

The application can then use ioctls to setup the FPGA as desired. Then it can call the **`poll()`** or **`select()`** system call to wait for status changes to become available. The application can use **`POLLPRI`** to watch for status changes (**`POLLIN`** is reserved for future use by customer applications which may need data collection notification, etc). If the driver indicates that the status changed then the application should read the status using the appropriate ioctl.

Once the application is done with the card it should **`close()`** the file handle.

The **ioctl()** interfaces available for the device driver are documented in an appendix at the end of this document. These interfaces are the primary means for controlling and getting status from the FPGA reference design. The amc520tool application utilizes these ioctls.

7.2 AMC520 Tool Application

The amc520tool provides basic support for controlling the FPGA and gathering status via the amc520_fpga.ko driver interfaces. It also supports various diagnostic tests.

To compile the tool application:

1) Enter the **tool** directory
2) Type **make**

The usage information for the tool is shown below:

```
AMC520 Tool v2.0.2 R1

usage: amc520tool <cmd> [<opts>]


LOW-LEVEL DEBUG CMDs:
        sig_test                 - Repeated read test
        scratch_test             - Repeated read/write test
        useriotest [0/1]         - Show/set Front User I/O test mode
        rtmtest [0/1]            - Show/set RTM Data/Clock LB test mode
        readreg <addr>           - Read BAR 4 PCIe register
        writereg <addr> <val>    - Write BAR 4 PCIe register
        dumpreg ['all']          - Dump BAR 4 PCIe contents
        readexreg <addr>         - Read BAR 5 PCIe register
        writeexreg <addr> <val>  - Write BAR 5 PCIe register
        dumpadc <chip>           - Dump ADC chip SPI registers
        readadc <chip> <addr>    - Read ADC chip SPI register byte
        writeadc <chip> <addr> <value>
                                 - Write ADC chip SPI register byte
        bpi_id                   - Report the FPGA BPI flash mfg/dev IDs
        bpi_dump [all]           - Dump the FPGA BPI flash contents
        bcistatus                - Wait for FPGA monitored status
        adcdelay <chip> [<tap>]  - Show/set ADC delay(chip=0-4, tap=0-31)
        dacdelay [<tap>]         - Show/set DAC delay (tap=0-31)
        syncdelay [<tap>]        - Show/set SYNC delay (tap=0-31)

PRIMARY CMDs:
        detect                   - Detect driver/FPGA versions
        portstatus ['refresh']   - Show FPGA backplane/SFP+ port statuses
        sysmon                   - Report system monitor info
        bpi_program <bin_file>   - Program the FPGA BPI flash
        dac [<xor_en> <chan0_mode> <chan1_mode>]
            chanX_mode=run_zeros|run_ones|run_mid|run_ramp|run_fsdiv2|
                       run_fsdiv4|run_fsdiv8|run_fsdiv16|(fixed value)|
                       run_adc<0-9>|stop
                                 - Show/set DAC mode
        routing [<target> <src>] - Show/Set FPGA clock routing target's source

          target: tclka, tclkb, tclkc, tclkd, trigstart, trigend, trigout
          src:    disable, zero, one, tclka, tclkb, tclkc, tclkd, trigstart,
                  trigend, trigin, dactoggle, adcclk, adc0toggle, adc1toggle,
                  adc2toggle, adc3toggle, adc4toggle
```

```
adcclksel [<sel>]        - Show/Set ADC clock selector

  sel:    rtmclk0, rtmclk1, rtmclk2, 125mhz, tclka, front

sync [<div>]             - Show/set ADC sync (div 0=OFF, 1=DIV2, ...)
adc <chan> <run_adc_analog|     - ADC analog samples
            run_adc_midscale|   - ADC fixed mid-scale
            run_adc_posfull|    - ADC fixed +full-scale
            run_adc_negfull|    - ADC fixed -full-scale
            run_adc_checker|    - ADC checkerboard
            run_adc_pnlong|     - ADC PRN long seq
            run_adc_pnshort|    - ADC PRN short seq
            run_adc_toggle|     - ADC toggle seq
            run_ramp|           - FPGA ramp pattern
            bist>               - Auto-BIST w/ ADC checkerboard
              - Operate an ADC channel with the given mode

NOTE: The stdout from this command is BINARY and should be redirected
      to a file or alternatively to a pipe such as netcat for
      direct transmission to another system. Only one ADC channel
      should be running at a time for this reference design.

      adccheck <filename>      - Check file for ADC overflow
      adcconv <filename>       - Convert binary file to CSV on stdout
```

The options for the tool generally follow the field definitions in the register specification. Please refer to the register specification later in the document for details. The **amc520_fpga.ko** driver module must be loaded prior to using the tool.

ADC Example (capturing channel 0 with on-board 125MHz clock):

The manual way of capturing data is shown below (a simpler way is shown following):

```
./amc520tool adc 0 run_adc_analog > /tmp/chan0.bin
./amc520tool adccheck /tmp/chan0.bin                    (optional)
./amc520tool adcconv /tmp/chan0.bin > /tmp/chan0.csv
```

The above example runs the ADC channel 0 and collects the snapshot data into a binary file. Then the binary file is optionally checked for overflow, etc. Finally the binary file is converted into a comma separated values file. Either file can be loaded into Matlab or some other analysis tool as desired. All ten channels can be checked one at a time.

A script called **capture** is provided to simplify the process shown above, so it can be condensed as:

```
./capture 0                    (outputs to chanX.bin and chanX.csv)
```

Another helper script called **capture_all** is provided which allows interactive capture of all 10 channels. The script prompts the user to press enter before each channel's data is captured so that the signal generation equipment may be setup, etc.

```
./capture_all                  (outputs to chan<0-9>.bin and chan<0-9>.csv)
```

DAC Examples:

```
amc520tool dac 1 run_fsdiv16 run_fsdiv16
```

The above example runs the DAC using XOR data mode with both channel 0 and channel 1 outputting a sine wave with frequency Fs/16.

```
amc520tool dac 1 run_adc3 run_adc4
```

The above example runs the DAC using XOR data mode with both channel DAC channel 0 outputting ADC channel 3's data and DAC channel 1 outputting ADC channel 4's data. This demonstrates the ADC-to-DAC loop-through feature.

Ethernet/PCIe Status Example:

```
amc520tool portstatus refresh
```

The above example shows the status of the SERDES ports on the FPGA including the two 1000Base-X ports to the AMC Ports 0 & 1, two PCIe x4 ports to AMC Ports 4-7 and 8-11, and two 1000Base-X ports to the front SFP+ ports. Press CTRL-C to exit the refresh loop.

Clock Routing Example:

```
amc520tool routing tclka dactoggle
amc520tool routing tclkb dactoggle
amc520tool routing tclkc dactoggle
amc520tool routing tclkd dactoggle
amc520tool routing trigstart dactoggle
amc520tool routing trigend dactoggle
amc520tool routing trigout dactoggle
```

The example above sends the 'dactoggle' signal (DAC clock divided by 2) to the four backplane TCLK M-LVDS channels, the two backplane M-LVDS trigger channels, and the front panel TRIG OUT connector.

NOTE: The DAC clock in this reference design is 125MHz resulting in a toggle signal of 62.5MHz.

8 Appendix A: FPGA PCIe Register Specification

The reference design's register space is controlled by an external CPU via PCIe x4 on AMC Ports 4-7. An external CPU may also connect via PCIe x4 on AMC Ports 8-11 however this port supports only link establishment and OS configuration but not internal registers.

8.1 FPGA Reference Design PCIe Config Space

The PCIe configuration space as seen by the external CPU attached to either AMC Ports 4-7 or AMC Ports 8-11:

Item	Value
Vendor ID	0xABCD (VadaTech Incorporated)
Device ID	0x4520 (AMC520)
Subsystem Vendor ID	0xABCD (VadaTech Incorporated)
Subsystem Device ID	0x4520 (AMC520)

Table 20: FPGA reference design PCIe device/vendor IDs

BAR	Size	Type	Access Style	Contents
4	64 KB	32-bit addr MMIO	32-bit PIO w/ byte enables	Internal registers
5	8 MB	32-bit addr MMIO	32-bit PIO w/ byte enables	(for customer expansion)

Table 21: FPGA reference design PCIe BAR configuration

The design supports a control/status register interface via BAR 4 using 32-bit PIO access to the internal registers.

The BAR 4 registers utilize a simple internal bus mechanism to interface to the PCIe Local Bridge core. This mechanism expects that read data is already available, and the local bridge will issue a read completion pulse once it is captured by the bridge (to facilitate popping a 'first word fall through' style FIFO for example). Write data is provided simultaneously with a write pulse, and the internal registers are expected to absorb the write data as fast as the local bridge can provide it. There is no flow control mechanism for the BAR 4 registers.

The BAR 5 registers utilize a handshaking mechanism intended to provide some means of flow control. This mechanism provides a single-clock read request pulse simultaneous with the address and then waits for a read acknowledge pulse. The read data is captured by the local bridge on the first clock cycle on which the read acknowledge pulse is seen by it. The read request will time-out after 10 clock cycles and will be abandoned if no acknowledgement is seen. This prevents lock-up of the PCIe bus. Write data is provided simultaneously with a write pulse, and the internal registers are expected to absorb the write data as fast as the local bridge can provide it. Therefore, there is a flow control mechanism for BAR 5 reads but not for writes.

NOTE: These BAR 4 & 5 access styles are provided simply as a reference. If your application requires a different access style, then please feel free to customize them to suit your needs. The goal of these mechanisms is simplicity and NOT performance. The PCIe interface can perform at greater efficiency if you design your registers/FIFOs to support PCIe pipelining and/or DMA operations. This is beyond the scope of the VadaTech reference design.

The design includes an interrupt controller which signals interrupts using either Legacy INTA or an MSI vector.

Refer to the register specification in a subsequent section for details of the internal registers BAR.

As mentioned previously, only the AMC Ports 4-7 PCIe actually has internal registers attached. The AMC Ports 8-11 PCIe is an exact clone of the first port but there are no internal registers attached to it. This second port is provided simply to verify the low-level hardware connectivity.

The register map and detailed register specification for AMC Ports 4-7 PCIe are shown on the following pages.

Core	BAR4 Offset	Mnemonic	Description
Interrupt Controller	0x0000	GIMSR	Global Interrupt Mask Set Register
	0x0004	GIMCR	Global Interrupt Mask Clear Register
	0x0008	GISR	Global Interrupt Status Register
	0x000C	BCSR	Bit-Change Status Register
	0x0010 – 0x00FF	N/A	Reserved
BPI Flash Controller	0x0100	BPICTRL	BPI Flash Control Register
	0x0104	BPIADDR	BPI Flash Address Register
	0x0108	BPIDATA	BPI Flash Data Register
	0x010C – 0x07FF	N/A	Reserved
Clock Router	0x0800	CREN	Clock Router Enables Register
	0x0804	CRTCLKA	Clock Router TCLKA Source Register
	0x0808	CRTCLKB	Clock Router TCLKB Source Register
	0x080C	CRTCLKC	Clock Router TCLKC Source Register
	0x0810	CRTCLKD	Clock Router TCLKD Source Register
	0x0814	CRTRIGSTART	Clock Router Trig Start Source Register
	0x0818	CRTRIGEND	Clock Router Trig End Source Register
	0x081C	CRTRIGOUT	Clock Router TRIG OUT Source Register
	0x0820 – 0x0FFF	N/A	Reserved
DAC	0x1000	DACCTRL	DAC Control Register
	0x1004	DACDELAY	DAC Delay Register
	0x1008	DACFD	DAC Fixed Data Register
	0x100C – 0x1FFF	N/A	Reserved
ADC Chip 0	0x2000	ADCCTRL0	ADC Chip 0 Control Register
	0x2004	ADCDELAY0	ADC Chip 0 Delay Register
	0x2008	ADCBIST00	ADC Chip 0 BIST 0 Register
	0x200C	ADCBIST10	ADC Chip 0 BIST 1 Register
	0x2010	SPICTRL0	ADC Chip 0 SPI Control Register
	0x2014 – 0x27FF	N/A	Rsvd
ADC Chip 1	0x2800	ADCCTRL1	ADC Chip 1 Control Register
	0x2804	ADCDELAY1	ADC Chip 1 Delay Register
	0x2808	ADCBIST01	ADC Chip 1 BIST 0 Register
	0x280C	ADCBIST11	ADC Chip 1 BIST 1 Register
	0x2810	SPICTRL1	ADC Chip 1 SPI Control Register
	0x2814 – 0x2FFF	N/A	Rsvd
ADC Chip 2	0x3000	ADCCTRL2	ADC Chip 2 Control Register
	0x3004	ADCDELAY2	ADC Chip 2 Delay Register
	0x3008	ADCBIST02	ADC Chip 2 BIST 0 Register
	0x300C	ADCBIST12	ADC Chip 2 BIST 1 Register
	0x3010	SPICTRL2	ADC Chip 2 SPI Control Register
	0x3014 – 0x37FF	N/A	Rsvd
ADC Chip 3	0x3800	ADCCTRL3	ADC Chip 3 Control Register
	0x3804	ADCDELAY3	ADC Chip 3 Delay Register
	0x3808	ADCBIST03	ADC Chip 3 BIST 0 Register
	0x380C	ADCBIST13	ADC Chip 3 BIST 1 Register
	0x3810	SPICTRL3	ADC Chip 3 SPI Control Register
	0x3814 – 0x3FFF	N/A	Rsvd

…continued on next page …

…continued from previous page…

ADC Chip 4	0x4000	ADCCTRL4	ADC Chip 4 Control Register
	0x4004	ADCDELAY4	ADC Chip 4 Delay Register
	0x4008	ADCBIST04	ADC Chip 4 BIST 0 Register
	0x400C	ADCBIST14	ADC Chip 4 BIST 1 Register
	0x4010	SPICTRL4	ADC Chip 4 SPI Control Register
	0x4014 – 0x47FF	N/A	Rsvd
ADC Synchronizer	0x4800	SYNCCTRL	ADC Synchronizer Control Register
	0x4804	SYNCDELAY	ADC Synchronizer Delay Register
	0x4808 – 0x4FFF	N/A	Rsvd
ADC Selector	0x5000	ADCSEL	ADC Selector Register
	0x5004	ADCDATA	ADC Data Register
	0x5008 – 0x6FFF	N/A	Rsvd
Sysmon	0x7000	TEMP	FPGA Temperature Register
	0x7004	VCCINT	FPGA Internal Voltage Register
	0x7008	VCCAUX	FPGA Auxiliary Voltage Register
	0x700C – 0x7FDF	N/A	Reserved
Utility	0x7FE0	BRDSTATUS	Board Status (including SFP+) Register
	0x7FE4	A3STATUS	AMC Port 3 (PCIe 8-11) Status Register
	0x7FE8	A2STATUS	AMC Port 2 (PCIe 4-7) Status Register
	0x7FEC	A1STATUS	AMC Port 1 (1000Base-X) Status Register
	0x7FF0	A0STATUS	AMC Port 0 (1000Base-X) Status Register
	0x7FF4	SCRATCH	Scratch Register
	0x7FF8	VER	FPGA Version Register
	0x7FFC	SIG	Signature Register

Table 22: FPGA reference design register map

The following pages describe the registers in detail. In these register descriptions certain field behavior tags are used. The following is a list of those behaviors:

RO	-	Read Only
R/W	-	Read/Write
R/W1C	-	Read/Write 1 to Clear
R/W1S	-	Read/Write 1 to Set

8.2 GIMSR – Global Interrupt Mask Set Register

Address: 0x0000

	31	30	29	28	27	26	25	24
Field	BCI	THERM	Rsvd					
Access	R/W1S	R/W1S	RO					
Reset	0	0	X					

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	Rsvd							
Access	RO							
Reset	X							

Bit(s)	Field	Description
31	BCI	**Read:** Bit Change Interrupt is enabled when '1' else it is disabled. **Write:** Writing '1' sets the Bit Change Interrupt enabled. Writing '0' has no effect.
30	THERM	**Read:** Thermal alert interrupt enabled when '1' else it is disabled. **Write:** Writing '1' sets the thermal alert interrupt enabled. Writing '0' has no effect.

8.3 GIMCR – Global Interrupt Mask Clear Register

Address: 0x0004

	31	30	29	28	27	26	25	24
Field	BCI	THERM	Rsvd					
Access	R/W1C	R/W1C	RO					
Reset	0	0	X					

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	Rsvd							
Access	RO							
Reset	X							

Bit(s)	Field	Description
31	BCI	**Read:** Bit Change Interrupt is enabled when '1' else it is disabled. **Write:** Writing '1' sets the Bit Change Interrupt disabled. Writing '0' has no effect.
30	THERM	**Read:** Thermal alert interrupt enabled when '1' else it is disabled. **Write:** Writing '1' sets the thermal alert interrupt disabled. Writing '0' has no effect.

8.4 GISR – Global Interrupt Status Register

Address: 0x0008

	31	30	29	28	27	26	25	24
Field	BCI	THERM	Rsvd					
Access	RO	RO	RO					
Reset	0	0	X					

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	Rsvd							
Access	RO							
Reset	X							

Bit(s)	Field	Description
31	BCI	Bit Change Interrupt is pending when '1' else it is not pending. Refer to BCSR register.
30	THERM	Thermal alert interrupt is pending when '1' else it is not pending. This status comes from an on-board ADT7461 thermal probe which monitors the FPGA's core temperature. This chip is configured by the MMC and is monitored by the MMC's IPMI application. The THERM indication to the FPGA is a secondary function of the chip.

NOTE: This register reflects the pending interrupt line statuses whether or not they are actually triggering an interrupt to the CPU. A status bit only triggers an actual interrupt to the CPU if its corresponding mask bit is also '1'.

There is no way to clear the interrupt status from this register. The interrupt must either be acknowledged at its source or masked within the interrupt controller to get the CPU interrupt to cease.

8.5 BCSR – Bit Change Status Register

Address: 0x000C

	31	30	29	28	27	26	25	24
Field	Rsvd	THERM	Rsvd					
Access	RO	R/W1C	RO					
Reset	X	0	X					

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	Rsvd							
Access	RO							
Reset	X							

Bit(s)	Field	Description
30	THERM	**Read:** The *THERM line on the board changed state when this bit is '1', else it did not change state. **Write:** Writing '1' clears this bit. Writing '0' has no effect.

8.6 BPICTRL - BPI Flash Control Register

Address: 0x0100

This register is used to control the BPI Flash interface in coordination with the BPIADDR and BPIDATA registers.

	31	30	29	28	27	26	25	24
Field	BUSY	Rsvd						
Access	RO	RO						
Reset	0	X						

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	Rsvd					RECONFIG	WRITE	EXEC
Access	RO					R/W1SC	R/W	R/W1SC
Reset	X					0	0	0

Bit(s)	Field	Description
31	BUSY	When this bit is '1' it indicates that the BPI Flash controller is busy. The software should not change the value of the BPI Flash controller registers during the busy period.
2	RECONFIG	This bit can be set to '1' to trigger a reconfiguration of the FPGA which in turn triggers the re-initialization of the MCU/GPS receiver.
1	WRITE	This bit should be set to '1' for a write operation and '0' for a read operation. The type of operation takes effect when the EXEC bit is set.
0	EXEC	When this bit is set to '1' it triggers execution of a BPI Flash read or write transaction as indicated by the WRITE bit.

WARNING: The PCIe upgrade capability assumes that a valid FPGA configuration file will be downloaded into the BPI Flash. If this mechanism is used in such a way that it results in the FPGA configuration image in the flash becoming corrupted or missing the AMC520 board will no longer function (including the PCIe upgrade mechanism) and the board will have to be programmed via JTAG.

To read:

1) Write the desired word address to the BPIADDR register.
2) Set the WRITE bit to zero and EXEC bit to one, then wait for the BUSY bit to clear.
3) Read the flash data from the BPIDATA:READ_DATA field.

To write:

1) Write the desired word address to the BPIADDR register.
2) Set the WRITE bit to one and EXEC bit to one, then wait for the BUSY bit to clear.

8.7 BPIADDR - BPI Flash Address Register

Address: 0x0104

This register is used to control the BPI Flash address lines.

0x8004	31	30	29	28	27	26	25	24
Field	Rsvd			ADDR[28:24]				
Access	RO			R/W				
Reset	X			0x00				

	23	22	21	20	19	18	17	16
Field	ADDR[23:16]							
Access	R/W							
Reset	0x00							

	15	14	13	12	11	10	9	8
Field	ADDR[15:8]							
Access	R/W							
Reset	0x00							

	7	6	5	4	3	2	1	0
Field	ADDR[7:0]							
Access	R/W							
Reset	0x00							

Bit(s)	Field	Description
0-28	ADDR	BPI Flash address to use for transaction.

8.8 BPIDATA - BPI Flash Data Register

Address: 0x0108

This register is used to control the BPI Flash data lines during a write and to receive the data from the flash on a read.

0x8008	31	30	29	28	27	26	25	24
Field	READ_DATA[15:8]							
Access	RO							
Reset	0x00							

	23	22	21	20	19	18	17	16
Field	READ_DATA[7:0]							
Access	RO							
Reset	0x00							

	15	14	13	12	11	10	9	8
Field	WRITE_DATA[15:8]							
Access	R/W							
Reset	0x00							

	7	6	5	4	3	2	1	0
Field	WRITE_DATA[7:0]							
Access	R/W							
Reset	0x00							

Bit(s)	Field	Description
16-31	READ_DATA	Holds the data captured by the FPGA during a read transaction.
0-15	WRITE_DATA	The data to be written to the FPGA should be stored here prior to executing the transaction.

8.9 CREN – Clock Routing Enable Register

Address: 0x0800

	31	30	29	28	27	26	25	24
Field	MLVDS_IN	Rsvd						
Access	R/W	RO						
Reset	0	X						

	23	22	21	20	19	18	17	16
Field	Rsvd					ADCCLKSEL		
Access	RO					R/W		
Reset	X					1		

	15	14	13	12	11	10	9	8
Field	Rsvd			USERIOTEST	Rsvd			RTMTEST
Access	RO			R/W	RO			R/W
Reset	X			0	X			0

	7	6	5	4	3	2	1	0
Field	Rsvd		TRIGEND_OUT	TRIGSTART_OUT	TCLKD_OUT	TCLKC_OUT	TCLKB_OUT	TCLKA_OUT
Access	RO		R/W	R/W	R/W	R/W	R/W	R/W
Reset	X		0	0	0	0	0	0

Bit(s)	Field	Description
0	TCLKA_OUT	Enable the TCLKA M-LVDS transmitter when '1' else disable it (receive).
1	TCLKB_OUT	Enable the TCLKB M-LVDS transmitter when '1' else disable it (receive).
2	TCLKC_OUT	Enable the TCLKC M-LVDS transmitter when '1' else disable it (receive).
3	TCLKD_OUT	Enable the TCLKD M-LVDS transmitter when '1' else disable it (receive).
4	TRIGSTART_OUT	Enable the TRIGSTART M-LVDS transmitter when '1' else disable it (receive).
5	TRIGEND_OUT	Enable the TRIGEND M-LVDS transmitter when '1' else disable it (receive).
8	RTMTEST	Enable the MRT520 RTM External loopback data/clock test when '1' else disable it. RTMTEST = D08 outputs 40MHz D09 outputs 20MHz D10 outputs 10MHz D11 outputs 5MHz (external loopback routes D08 to D04) (external loopback routes D09 to D05) (external loopback routes D10 to D06) (external loopback routes D11 to D07) D00 outputs 40MHz received from D04 D01 outputs 20MHz received from D05 D02 outputs 10MHz received from D06 D03 outputs 5MHz received from D07 (external loopback routes D00 to CLK2) (external loopback routes D01 to CLK0) (external loopback routes D02 to CLK1) (external loopback routes D03 to unused TTL.C on connector)

12	USERIOTEST	Enable the front panel USER I/O test when '1' else disable it. USERIOTEST = USERIO[15 downto 0] outputting from board counting up at 80MHz
18-16	ADCCLKSEL	Selects the source for the ADC clock: 0: RTM CLK0 1: On-board 125MHz 2: Reserved (stop clock) 3: Reserved (stop clock) 4: Backplane TCLKA 5: RTM CLK1 6: RTM CLK2 7: Front CLK IN
31	MLVDS_IN	Enable all the backplane M-LVDS receivers when '1' else disable them. NOTE: It is OK to have the receivers enabled all the time even when transmitting on some of the clock channels. But there should usually only be one transmitter on any given backplane clock channel (i.e. choose MCH or AMC to drive each channel). The software in the reference design only enables the M-LVDS receivers if one of the backplane clocks/triggers is actually being used as the source of a clock route; this may help to minimize the noise level of the board for ADC capture.

8.10 CRTCLKA/B/C/D/TRIGSTART/TRIGEND/TRIGOUT – Clock Routing Registers

Addresses: 0x0804 (CRTCLKA), 0x0808 (CRTCLKB), 0x080C (CRTCLKC), 0x0810 (CRTCLKD), 0x0814 (CRTRIGSTART), 0x0818 (CRTRIGEND), 0x081C (CRTRIGOUT)

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	Rsvd				SOURCE			
Access	RO				R/W			
Reset	X				(see below)			

Bit(s)	Field	Description
3-0	SOURCE	This field selects the source signal to drive to the destination as follows: 0x0: Fixed logic '0' value 0x1: Fixed logic '1' value 0x2: Backplane TCLKA input 0x3: Backplane TCLKB input 0x4: Backplane TCLKC input 0x5: Backplane TCLKD input 0x6: Backplane TRIGSTART input 0x7: Backplane TRIGEND input 0x8: Front panel TRIG IN input 0x9: DAC Toggle Bit (half frequency of DAC fabric clock) 0xA: ADC Global Clock input 0xB: ADC0 Toggle Bit (half frequency of ADC0 fabric clock) 0xC: ADC1 Toggle Bit (half frequency of ADC1 fabric clock) 0xD: ADC2 Toggle Bit (half frequency of ADC2 fabric clock) 0xE: ADC3 Toggle Bit (half frequency of ADC3 fabric clock) 0xF: ADC4 Toggle Bit (half frequency of ADC4 fabric clock)

The Clock Routing Registers all follow the same layout. The registers correspond to MUX selectors in the FPGA and select the source signal for a given target. The targets are as follows:

Register	For designating signal to route to	Reset default
CRTCLKA (0x0804)	Backplane TCLKA output	0 (low)
CRTCLKB (0x0808)	Backplane TCLKB output	0 (low)
CRTCLKC (0x080C)	Backplane TCLKC output	0 (low)
CRTCLKD (0x0810)	Backplane TCLKD output	0 (low)
CRTRIGSTART (0x0814)	Backplane TRIGSTART output	0 (low)
CRTRIGEND (0x0818)	Backplane TRIGEND output	0 (low)
CRTRIGOUT (0x081C)	Front panel TRIG OUT	8 (TRIG IN loopback)

NOTE: A given backplane clock input cannot source its own output.

8.11 DACCTRL – DAC Control Register

Address: 0x1000

	31	30	29	28	27	26	25	24
Field	RESET	XOR_EN	TORB	PD	Rsvd			
Access	R/W	R/W	R/W	R/W	RO			
Reset	1	1	0	1	X			

	23	22	21	20	19	18	17	16
Field	Rsvd							TOGGLE
Access	RO							RO
Reset	X							0

	15	14	13	12	11	10	9	8
Field	Rsvd			GEN1				
Access	RO			R/W				
Reset	X			0x00				

	7	6	5	4	3	2	1	0
Field	Rsvd			GEN0				
Access	RO			R/W				
Reset	X			0x00				

Bit(s)	Field	Description
0-4	GEN0	This field selects the DAC channel 0 data output pattern: 0x00: Generate all zero data values 0x01: Generate all ones data values 0x02: Generate mid-level data value 0x03: Generate ramp data values 0x04: Generate Fs/2 Sine data values 0x05: Generate Fs/4 Sine data values 0x06: Generate Fs/8 Sine data values 0x07: Generate Fs/16 Sine data values 0x08: Generate fixed data value from the DACFD register 0x09 – 0x0F: Reserved 0x10: Loop-through the ADC channel 0 data to the DAC 0x11: Loop-through the ADC channel 1 data to the DAC 0x12: Loop-through the ADC channel 2 data to the DAC 0x13: Loop-through the ADC channel 3 data to the DAC 0x14: Loop-through the ADC channel 4 data to the DAC 0x15: Loop-through the ADC channel 5 data to the DAC 0x16: Loop-through the ADC channel 6 data to the DAC 0x17: Loop-through the ADC channel 7 data to the DAC 0x18: Loop-through the ADC channel 8 data to the DAC 0x19: Loop-through the ADC channel 9 data to the DAC 0x1A – 0x1F: Reserved
8-12	GEN1	This field selects the DAC channel 1 data output pattern (refer to GEN0 for possible values).
16	TOGGLE	This bit toggles when the DAC clock is running.
28	PD	Power down the DAC chip when '1' else power it up.

29	TORB	Select Offset Binary mode when '1', else select Two's Complement.
30	XOR_EN	Enable the XOR function on the DAC data outputs when '1', else don't use XOR function.
31	RESET	Reset the DAC cores when '1', else release them from reset.

8.12 DACDELAY – DAC Delay Register

Address: 0x1004

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	Rsvd			TAP				
Access	RO			R/W				
Reset	X			0				

Bit(s)	Field	Description
0-4	TAP	This field selects the ODELAY tap to use for the FPGA DAC data outputs. It can be used to adjust the relative phase relationship between the system-synchronous DAC clock and the DAC data to enhance the data setup/hold time at the DAC chip's inputs. Refer to Xilinx ODELAY documentation for further details. NOTE: This value should only be changed when the DACCTRL:RESET bit is '1'.

8.13 DACFD – DAC Fixed Data Register

Address: 0x1008

	31	30	29	28	27	26	25	24
Field	FD1[15:8]							
Access	R/W							
Reset	(see below)							

	23	22	21	20	19	18	17	16
Field	FD1[7:0]							
Access	R/W							
Reset	0x0000							

	15	14	13	12	11	10	9	8
Field	FD0[15:8]							
Access	R/W							
Reset	(see below)							

	7	6	5	4	3	2	1	0
Field	FD0[7:0]							
Access	R/W							
Reset	0x0000							

Bit(s)	Field	Description
0-15	FD0	Fixed data value to provide to DAC channel 0 when DACCTRL:GEN0=0x8.
16-31	FD1	Fixed data value to provide to DAC channel 1 when DACCTRL:GEN1=0x8.

8.14 ADCCTRL0/1/2/3/4 – ADC Chip 0/1/2/3/4 Control Registers

Address: 0x2000, 0x2800, 0x3000, 0x3800, 0x4000

	31	30	29	28	27	26	25	24
Field	RESET	PDWN	OE	Rsvd				
Access	R/W	R/W	R/W	RO				
Reset	1	0	1	X				

	23	22	21	20	19	18	17	16
Field	Rsvd							TOGGLE
Access	RO							RO
Reset	X							0

	15	14	13	12	11	10	9	8
Field	OVERFLOW1	Rsvd					BIST_RESET1	RAMP_GEN1
Access	RO	RO					R/W	R/W
Reset	0	X					1	0

	7	6	5	4	3	2	1	0
Field	OVERFLOW0	Rsvd					BIST_RESET0	RAMP_GEN0
Access	RO	RO					R/W	R/W
Reset	0	X					1	0

Bit(s)	Field	Description
0	RAMP_GEN0	Generate internal ramp data on ADC channel A when '1', else pass through the actual ADC samples.
1	BIST_RESET0	Reset the BIST verifier for ADC channel A when '1', else let it run.
7	OVERFLOW0	When this bit is '1' it indicates that the clock domain crossing FIFO between the ADC Channel A and the ADC Selector core overflowed. This flag can be cleared with the RESET bit.
8	RAMP_GEN1	Generate internal ramp data on ADC channel B when '1', else pass through the actual ADC samples.
9	BIST_RESET1	Reset the BIST verifier for ADC channel B when '1', else let it run.
15	OVERFLOW1	When this bit is '1' it indicates that the clock domain crossing FIFO between the ADC Channel B and the ADC Selector core overflowed. This flag can be cleared with the RESET bit.
16	TOGGLE	This bit toggles when the ADC source-synchronous chip clock is running.
29	OE	This bit enables the ADC chip's data outputs when '1' else it disables them.
30	PDWN	When '1' power-down the ADC chip, else power it up.
31	RESET	When '1' reset the ADC cores for this chip, else let them run.

8.15 ADCDELAY0/1/2/3/4 – ADC Chip 0/1/2/3/4 Delay Registers

Address: 0x2004, 0x2804, 0x3004, 0x3804, 0x4004

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	Rsvd			TAP				
Access	RO			R/W				
Reset	X			0				

Bit(s)	Field	Description
0-4	TAP	This field selects the IDELAY tap to use for the FPGA ADC data inputs. It can be used to adjust the relative phase relationship between the source-synchronous ADC clock and the ADC data to enhance the data setup/hold time at the FPGA's inputs. Refer to Xilinx IDELAY documentation for further details. NOTE: This value should only be changed when the ADCCTRLx:RESET bit is '1'.

8.16 ADCBIST[0/1]0/1/2/3/4 – ADC Chip 0/1/2/3/4 BIST 0/1 Registers

Address: 0x2008/0x200C, 0x2808/0x280C, 0x3008/0x300C, 0x3808/0x380C, 0x4008/0x400C

	31	30	29	28	27	26	25	24
Field	ERROR	STARTED	Rsvd					
Access	RO	RO	RO					
Reset	0	0	X					

	23	22	21	20	19	18	17	16
Field	Rsvd							WORD[16]
Access	RO							RO
Reset	X							0

	15	14	13	12	11	10	9	8
Field	WORD[15:8]							
Access	RO							
Reset	0							

	7	6	5	4	3	2	1	0
Field	WORD[7:0]							
Access	RO							
Reset	0							

Bit(s)	Field	Description
0-16	WORD	This field captures the data from the ADC channel. If the BIST is in reset this field will simply reflect the current sample from the ADC. If the BIST is not in reset then this field will reflect the current sample from the ADC until an error occurs at which point it will hold onto the sample that caused the error so that it can be inspected. NOTE: This field may show unpredictable results if the value is changing when it is read. It is intended to be read only after the ERROR flag is set and the value is stable.
30	STARTED	This field shows '0' until the BIST verifier starts running for the given channel at which point it becomes '1'.
31	ERROR	This field shows '0' until a BIST verifier error occurs for the given channel at which point it becomes '1'.

8.17 SPICTRL0/1/2/3/4 – ADC Chip 0/1/2/3/4 SPI Control Register

Address: 0x2010, 0x2810, 0x3010, 0x3810, 0x4010

	31	30	29	28	27	26	25	24
Field	RESET	DONE	RNOTW	ADDR[12:0]				
Access	R/W	RO	R/W	R/W				
Reset	1	0	1	(see below)				

	23	22	21	20	19	18	17	16
Field	ADDR[7:0]							
Access	R/W							
Reset	0x0000							

	15	14	13	12	11	10	9	8
Field	READ_DATA							
Access	RO							
Reset	0x00							

	7	6	5	4	3	2	1	0
Field	WRITE_DATA							
Access	R/W							
Reset	0x00							

Bit(s)	Field	Description
0-7	WRITE_DATA	Data to write to the ADC chip via SPI is placed into this field prior to releasing the controller from reset.
8-15	READ_DATA	Data read from the ADC chip via SPI is placed into this field after the DONE bit is set by the controller.
16-28	ADDR	The address to read/write via SPI is placed into this field prior to releasing the controller from reset.
29	RNOTW	This flag indicates a READ operation when '1', else a WRITE operation.
30	DONE	This flag indicates that the requested operation has completed when '1', else it has not completed yet.
31	RESET	When this field is '1' the SPI controller is held in reset, else it is allowed to run. Setup the other fields before releasing the core from reset as the controller performs the transaction immediately upon being released from reset.

8.18 SYNCCTRL – ADC Synchronizer Control Register

Address: 0x4800

	31	30	29	28	27	26	25	24
Field	RESET	Rsvd						
Access	R/W	RO						
Reset	1	X						

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	DIV							
Access	R/W							
Reset	0x00							

Bit(s)	Field	Description
0-7	DIV	This field sets the divisor for the SYNC pulse generation to the ADC chips (which is generated by dividing down the ADC global clock). If this field is set to 0x00 or the RESET bit is set then SYNC pulse generation is disabled. Otherwise the SYNC pulse is the clock divided by (DIV + 1).
31	RESET	When this field is '1' the ADC Sychronizer core is held in reset, otherwise it is allowed to run.

8.19 SYNCDELAY – ADC Synchronizer Delay Register

Address: 0x4804

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	Rsvd			TAP				
Access	RO			R/W				
Reset	X			0				

Bit(s)	Field	Description
0-4	TAP	This field selects the ODELAY tap to use for the FPGA SYNC pulse outputs. It can be used to adjust the relative phase relationship between the system-synchronous ADC global clock and the SYNC pulses to enhance the pulse's setup/hold time at the ADC chip inputs. Refer to Xilinx ODELAY documentation for further details. NOTE: This value should only be changed when the SYNCCTRL:RESET bit is '1'.

8.20 ADCSEL – ADC Selector Register

Address: 0x5000

	31	30	29	28	27	26	25	24
Field	RESET	OVERFLOW	UNDERFLOW	EMPTY	CHAN			
Access	R/W	RO	RO	RO	R/W			
Reset	1	0	0	1	0x0			

	23	22	21	20	19	18	17	16
Field	Rsvd							READ_AVAIL[16]
Access	RO							RO
Reset	X							(see below)

	15	14	13	12	11	10	9	8
Field	READ_AVAIL[15:8]							
Access	RO							
Reset	(see below)							

	7	6	5	4	3	2	1	0
Field	READ_AVAIL[7:0]							
Access	RO							
Reset	0x00000							

Bit(s)	Field	Description
0-16	READ_AVAIL	This field indicates the number of FIFO entries available to be read in the ADC Selector storage FIFO.
24-27	CHAN	This field selects which ADC channel is desired as the input to the ADC Selector storage FIFO in the range 0-9 (other values reserved).
28	EMPTY	This field shows as '0' when there is at least one entry in the storage FIFO, else it shows '1'.
29	UNDERFLOW	This field shows '0' normally but will show '1' if the FIFO is attempted to be read when it is empty.
30	OVERFLOW	This field shows '0' until the FIFO overflows at which point it shows '1'. No data corruption occurs when the FIFO overflows and this can be used as normal mechanism to know when the FIFO is full of data to read. Generally the sequence should be: 1) Configure the ADC Selector and release it from reset 2) Configure the selected ADC Chip/Channel and release them from reset 3) Wait for this OVERFLOW bit to be set 4) Reset the ADC Chip/Channel to keep it from providing more samples 5) Read out the entire contents of the FIFO
31	RESET	When this field is '1' it holds the ADC Selector core in reset, else it is allowed to run.

8.21 ADCDATA – ADC Data Register

Address: 0x5004

	31	30	29	28	27	26	25	24
Field	SAMPLE0[15:8]							
Access	RO							
Reset	(see below)							

	23	22	21	20	19	18	17	16
Field	SAMPLE0[7:0]							
Access	RO							
Reset	0x0000							

	15	14	13	12	11	10	9	8
Field	SAMPLE1[15:8]							
Access	RO							
Reset	(see below)							

	7	6	5	4	3	2	1	0
Field	SAMPLE1[7:0]							
Access	RO							
Reset	0x0000							

Bit(s)	Field	Description
0-15	SAMPLE1	This field holds the second sample of the current ADC Selector storage FIFO entry. **NOTE:** Reading this field causes the current FIFO entry (2 samples) to pop off of the FIFO to prepare the next entry to be read.
16-31	SAMPLE0	This field holds the first sample of the current ADC Selector storage FIFO entry.

8.22 TEMP – FPGA Temperature Register

Address: 0x7000

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	TEMP[15:8]							
Access	RO							
Reset	0							

	7	6	5	4	3	2	1	0
Field	TEMP[7:0]							
Access	RO							
Reset	0							

Bit(s)	Field	Description
0-15	TEMP	This field provides the internal die temperature reading of the FPGA. Refer to the Xilinx UG370 document for important details. **NOTE:** Similar information is available via chassis IPMI mechanisms which utilize the on-board AMC MMC CPU to read the temperature/voltage external to the FPGA.

8.23 VCCINT – FPGA Internal Voltage Register

Address: 0x7004

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	VCCINT[15:8]							
Access	RO							
Reset	0							

	7	6	5	4	3	2	1	0
Field	VCCINT[7:0]							
Access	RO							
Reset	0							

Bit(s)	Field	Description
0-15	VCCINT	This field provides the internal voltage reading of the FPGA. Refer to the Xilinx UG370 document for important details. **NOTE:** Similar information is available via chassis IPMI mechanisms which utilize the on-board AMC MMC CPU to read the temperature/voltage external to the FPGA.

8.24 VCCAUX – FPGA Auxiliary Voltage Register

Address: 0x7008

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	VCCAUX[15:8]							
Access	RO							
Reset	0							

	7	6	5	4	3	2	1	0
Field	VCCAUX[7:0]							
Access	RO							
Reset	0							

Bit(s)	Field	Description
0-15	VCCAUX	This field provides the auxiliary voltage reading of the FPGA. Refer to the Xilinx UG370 document for important details. NOTE: Similar information is available via chassis IPMI mechanisms which utilize the on-board AMC MMC CPU to read the temperature/voltage external to the FPGA.

8.25 BRDSTATUS – Board Status Register

Address: 0x7FE0

	31	30	29	28	27	26	25	24
Field	SDA	SCL	DACINOK	DACLOCKED	Rsvd			WP
Access	RO	RO	RO	RO	RO			RO
Reset	X	X	X	X	X			X

	23	22	21	20	19	18	17	16
Field	Rsvd		DENSITY		Rsvd		SPEED	
Access	RO		RO		RO		RO	
Reset	X		(depends on FPGA project)		X		(depends on FPGA project)	

	15	14	13	12	11	10	9	8
Field	Rsvd					SFP1_TXFLT	SFP1_RXLOS	SFP1_SYNC
Access	RO					RO	RO	RO
Reset	X					X	X	X

	7	6	5	4	3	2	1	0
Field	Rsvd					SFP0_TXFLT	SFP0_RXLOS	SFP0_SYNC
Access	RO					RO	RO	RO
Reset	X					X	X	X

Bit(s)	Field	Description
0/8	SFP0/1_SYNC	When '1' indicates that the corresponding SFP port is SYNCed, else not. (Reported by FPGA 1000Base-X IP)
1/9	SFP0/1_RXLOS	When '1' indicates a loss of signal condition on the corresponding SFP port, else an incoming signal was detected. (Reported by SFP module)
2/10	SFP0/1_TXFLT	When '1' indicates a transmit fault condition on the corresponding SFP port, else no transmit fault. (Reported by SFP module)
16-17	SPEED	This value is specified as a GENERIC in the FPGA project settings that generated the FPGA image. The value is defined as follows: 0: Reserved 1: Speed grade -1 2: Speed grade -2 3: Reserved
20-21	DENSITY	This value is specified as a GENERIC in the FPGA project settings that generated the FPGA image. The value is defined as follows: 0: LX240T 1: LX365T 2: LX550T 3: SX475T
24	WP	Indicates that the on-board flash is write protected when '1' else it is not.
28	DACLOCKED	When '1' indicates that the DAC MMCM is locked to the ADC input clock, else it is not. **NOTE:** The MMCM expects 125MHz at its input (such as the on-board 125MHz).

29	DACINOK	When ‘1’ indicates that the DAC MMCM’s input appears to be running, else it appears to have stopped.
30	SCL	Shows the current state of the I2C bus SCL line. (placeholder)
31	SDA	Shows the current state of the I2C bus SDA line. (placeholder)

8.26 A0/1STATUS – AMC Port 0/1 Status Registers

Address: 0x7FF0, 0x7FEC

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	Rsvd							1G_SYNC0
Access	RO							RO
Reset	X							0

Bit(s)	Field	Description
0	1G_SYNC0	AMC Lane 0/1 SYNCed when '1' else not SYNCed.

A0STATUS (0x7FF0): 1000Base-X to AMC Port 0
A1STATUS (0x7FEC): 1000Base-X to AMC Port 1

8.27 A2/3STATUS – AMC Ports 4-7/8-11 Status Registers

Address: 0x7FE8, 0x7FE4

	31	30	29	28	27	26	25	24
Field	PCIE_LINK	Rsvd	PCIE_WIDTH					
Access	RO	RO	RO					
Reset	0	X	0x00					

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	Rsvd							
Access	RO							
Reset	X							

Bit(s)	Field	Description
24-29	PCIE_WIDTH	Indicates the currently linked width of the backplane PCIe bus: 0x01: PCIe x1 0x02: PCIe x2 0x04: PCIe x4 0x08: PCIe x8 Others: Reserved
31	PCIE_LINK	PCIe bus linked when '1' else not linked.

A2STATUS (0x7FE8): PCIE to AMC Ports 4-7 (PCIe x4 to MCH 1 – Register interface)
A3STATUS (0x7FE4): PCIE to AMC Ports 8-11 (PCIe x4 to MCH 2 – Config only)

8.28 SCRATCH – Scratch Register

Address: 0x7FF4

	31	30	29	28	27	26	25	24
Field	SCRATCH[31:24]							
Access	R/W							
Reset	(see below)							

	23	22	21	20	19	18	17	16
Field	SCRATCH[23:16]							
Access	R/W							
Reset	(see below)							

	15	14	13	12	11	10	9	8
Field	SCRATCH[15:8]							
Access	R/W							
Reset	(see below)							

	7	6	5	4	3	2	1	0
Field	SCRATCH[7:0]							
Access	R/W							
Reset	0x00000000							

Bit(s)	Field	Description
31-0	SCRATCH	Scratchpad area for PCIe bus testing or free for other software use.

8.29 VER – FPGA Version Register

Address: 0x7FF8

	31	30	29	28	27	26	25	24
Field	MAJOR							
Access	RO							
Reset	(varies based on FPGA release)							

	23	22	21	20	19	18	17	16
Field	MINOR							
Access	RO							
Reset	(varies based on FPGA release)							

	15	14	13	12	11	10	9	8
Field	PATCH							
Access	RO							
Reset	(varies based on FPGA release)							

	7	6	5	4	3	2	1	0
Field	REV							
Access	RO							
Reset	(varies based on FPGA release)							

Bit(s)	Field	Description
31-24	MAJOR	Major release number of the FPGA image.
23-16	MINOR	Minor release number of the FPGA image.
15-8	PATCH	Patch release number of the FPGA image.
7-0	REV	Revision release number of the FPGA image.

The version number is printed as follows: v<MAJOR>.<MINOR>.<PATCH> R<REVISION>. For example: v1.0.0 R3.

8.30 SIG –Signature Register

Address: 0x7FFC

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	SIG							
Access	RO							
Reset	0x20							

Bit(s)	Field	Description
7-0	SIG	Unchanging value which helps to verify that the proper FPGA image is programmed into the part.

9 Appendix B: Linux Device Driver IOCTL Spec

The interfaces to the driver are exported in the amc520_fpga.h header file which should be included in any user application that uses the driver.

9.1 AMC520_IOC_GET_INFO

```
typedef struct {
        unsigned char major;
        unsigned char minor;
        unsigned char patch;
        unsigned char rev;
} amc520_version_t;

#define AMC520_NAME_LEN 20      /* amc520_fpga-BB.DD.F\0 */

typedef enum {
        AMC520_FPGA_LX240T = 0,
        AMC520_FPGA_LX365T,
        AMC520_FPGA_LX550T,
        AMC520_FPGA_SX475T
} amc520_fpga_density_t;

// This structure contains information that is not expected to change during
runtime
typedef struct {
        amc520_version_t        driver_version;

        unsigned                fpga_identified   : 1;  /* 0=only
driver_version valid, 1=all valid */
        amc520_version_t        fpga_version;
        amc520_fpga_density_t   fpga_density;
        unsigned                fpga_speed        : 2;
        unsigned                flash_wp          : 1;
} amc520_info_t;
```

Usage:

```
      int fd;
      amc520_info_t info;
      ioctl( fd, AMC520_IOC_GET_INFO, &info );
```

This call returns information about the driver/FPGA.

9.2 AMC520_IOC_GET_BCISTATUS

```
typedef struct {
        unsigned                therm           : 1;
} amc520_bcistatus_t;
```

Usage:

```
      int fd;
      amc520_bcistatus_t bcistatus;
      ioctl( fd, AMC520_IOC_GET_BCISTATUS, &bcistatus );
```

This call returns BCI status from the FPGA. The application may use the select() or poll() call using POLLPRI to monitor this structure for changes. The FPGA interrupts the CPU any time the structure changes and the device driver will wake the sleeping application process so that it can then read the status.

9.3 AMC520_IOC_GET/SET_CLOCK_ROUTING

```
#define AMC520_SOURCE_ZERO              0x00
#define AMC520_SOURCE_ONE               0x01
#define AMC520_SOURCE_TCLKA             0x02
#define AMC520_SOURCE_TCLKB             0x03
#define AMC520_SOURCE_TCLKC             0x04
#define AMC520_SOURCE_TCLKD             0x05
#define AMC520_SOURCE_TRIGSTART         0x06
#define AMC520_SOURCE_TRIGEND           0x07
#define AMC520_SOURCE_TRIGIN            0x08
#define AMC520_SOURCE_DACTOGGLE         0x09
#define AMC520_SOURCE_ADCCLK            0x0A
#define AMC520_SOURCE_ADC0TOGGLE        0x0B
#define AMC520_SOURCE_ADC1TOGGLE        0x0C
#define AMC520_SOURCE_ADC2TOGGLE        0x0D
#define AMC520_SOURCE_ADC3TOGGLE        0x0E
#define AMC520_SOURCE_ADC4TOGGLE        0x0F

#define AMC520_ADCCLKSEL_FROM_RTMCLK0           0
#define AMC520_ADCCLKSEL_FROM_ONBOARD125MHZ     1
#define AMC520_ADCCLKSEL_FROM_RESERVED2         2
#define AMC520_ADCCLKSEL_FROM_RESERVED3         3
#define AMC520_ADCCLKSEL_FROM_TCLKA             4
#define AMC520_ADCCLKSEL_FROM_RTMCLK1           5
#define AMC520_ADCCLKSEL_FROM_RTMCLK2           6
#define AMC520_ADCCLKSEL_FROM_FRONTCLKIN        7

typedef struct {
        /* clock routes */
        unsigned        tclka_out       : 4;
        unsigned        tclkb_out       : 4;
        unsigned        tclkc_out       : 4;
        unsigned        tclkd_out       : 4;
        unsigned        trigstart_out   : 4;
        unsigned        trigend_out     : 4;
        unsigned        trigout_out     : 4;

        /* path enables */
        unsigned        mlvds_in_en       : 1;
        unsigned        tclka_out_en      : 1;
        unsigned        tclkb_out_en      : 1;
        unsigned        tclkc_out_en      : 1;
        unsigned        tclkd_out_en      : 1;
        unsigned        trigstart_out_en  : 1;
        unsigned        trigend_out_en    : 1;

        /* test enables */
        unsigned        useriotest        : 1;
        unsigned        rtmtest           : 1;

        /* board-level clock routing selectors */
        unsigned        adcclksel         : 3;
} amc520_clock_routing_t;
```

Usage:

```
int fd;
amc520_clock_routing_t routing;
ioctl( fd, AMC520_IOC_GET_CLOCK_ROUTING, &routing );
ioctl( fd, AMC520_IOC_SET_CLOCK_ROUTING, &routing );
```

These calls get/set the Clock Router registers.

9.4 AMC520_IOC_GET_PORTSTATUS

```
typedef struct {
        unsigned sync    : 1;
} amc520_1g_portstatus_t;

typedef struct {
      unsigned tx_fault : 1;
      unsigned rxlos    : 1;
      unsigned sync     : 1;
} amc520_sfp_portstatus_t;

typedef struct {
        unsigned link    : 1;
        unsigned width   : 6;
} amc520_pcie_portstatus_t;

typedef struct {
      amc520_1g_portstatus_t   amc_1g[2];
      amc520_pcie_portstatus_t amc_pcie[2];
      amc520_sfp_portstatus_t  sfp[2];
} amc520_portstatus_t;
```

Usage:

```
int fd;
amc520_portstatus_t portstatus;
ioctl( fd, AMC520_IOC_GET_PORTSTATUS, &portstatus );
```

This call returns status information about the backplane interfaces of the FPGA.

9.5 AMC520_IOC_GET/SET_REG

```
typedef struct {
        unsigned int          offset;
        unsigned int          value;
} amc520_reg_t;
```

Usage:

```
      int fd;
      amc520_reg_t reg;
      ioctl( fd, AMC520_IOC_GET_REG, &reg );
      ioctl( fd, AMC520_IOC_SET_REG, &reg );
```

This call reads or writes a BAR 4 PCIe register in the FPGA.

9.6 AMC520_IOC_GET/SET_EXREG

```
typedef struct {
        unsigned int          offset;
        unsigned int          value;
} amc520_reg_t;
```

Usage:

```
      int fd;
      amc520_reg_t reg;
      ioctl( fd, AMC520_IOC_GET_EXREG, &reg );
      ioctl( fd, AMC520_IOC_SET_EXREG, &reg );
```

This call reads or writes a BAR 5 PCIe register in the FPGA.

9.7 AMC520_IOC_GET_SYSMON

```
typedef struct {
        unsigned short temp;
        unsigned short vccint;
        unsigned short vccaux;
} amc520_sysmon_t;
```

Usage:

```
      int fd;
      amc520_sysmon_t sysmon;
      ioctl( fd, AMC520_IOC_GET_SYSMON, &sysmon );
```

This call gets the FPGA's SYSMON A-to-D values for reporting temperatures and voltages.

9.8 AMC520_IOC_GET/SET_ADCCTRL

```
#define AMC520_ADC_CHANS_PER_CHIP 2
#define AMC520_ADC_CHIPS          5
#define AMC520_ADC_CHANS         (AMC520_ADC_CHANS_PER_CHIP *
AMC520_ADC_CHIPS)

typedef struct {
        unsigned        overflow   : 1;   // ADC channel output FIFO overflow
(read-only)
        unsigned        ramp_gen   : 1;   // FPGA Ramp pattern instead of ADC
data
        unsigned        bist_reset : 1;   // BIST error status reset
        unsigned        bist_error : 1;   // BIST error occurred (read-only)
        unsigned        bist_started : 1; // BIST started running (read-only)
        unsigned        bist_word  : 17;  // BIST word data capture (read-
only)
} amc520_adcchan_t;

typedef struct {
        unsigned int     chip_idx;        // selects chip for get/set

        unsigned         reset         : 1;
        unsigned         power_down    : 1;
        unsigned         output_enable : 1;
        unsigned         toggle        : 1;             // read-only
        amc520_adcchan_t chan[AMC520_ADC_CHANS_PER_CHIP];
} amc520_adcctrl_t;
```

Usage:

```
      int fd;
      amc520_adcctrl_t adcctrl;
      ioctl( fd, AMC520_IOC_GET_ADCCTRL, &adcctrl );
      ioctl( fd, AMC520_IOC_SET_ADCCTRL, &adcctrl );
```

These calls get or set the ADC control parameters.

9.9 AMC520_IOC_GET/SET_ADCSEL

```
typedef struct {
        unsigned         reset           : 1;
        unsigned         chan            : 4;

        unsigned         overflow        : 1;   // read-only
        unsigned         underflow       : 1;   // read-only
        unsigned         empty           : 1;   // read-only

        unsigned int     read_available;        // read-only
} amc520_adcsel_t;
```

Usage:

```
      int fd;
      amc520_adcsel_t adcsel;
      ioctl( fd, AMC520_IOC_GET_ADCSEL, &adcsel );
      ioctl( fd, AMC520_IOC_SET_ADCSEL, &adcsel );
```

These calls get/set the ADC selector control/status information.

9.10 AMC520_IOC_GET_ADCDATA

```
typedef struct {
        unsigned int      buffer_entries;          // # samples to put into
buffer
        unsigned short*   buffer;                   // user-space buffer for
driver to put data into
} amc520_adcdata_t;
```

Usage:

```
      int fd;
      amc520_adcdata_t adcdata;
      ioctl( fd, AMC520_IOC_GET_ADCDATA, &adcdata );
```

This call return data from the ADC Selector's storage FIFO in the FPGA.

9.11 AMC520_IOC_GET/SET_ADCREG

```
typedef struct {
        unsigned int    chip_idx;

        unsigned        addr  : 13;
        unsigned char   value;
} amc520_adcreg_t;
```

Usage:

```
      int fd;
      amc520_adcreg_t adcreg;
      ioctl( fd, AMC520_IOC_SET_ADCREG, &adcreg );
```

This call uses the serial controller within the FPGA to get/set an ADC register.

9.12 AMC520_IOC_GET/SET_SYNCCTRL

```
typedef struct {
        unsigned         reset         : 1;
        unsigned         div           : 8; // 0=Disabled, 1=DIV2, 2=DIV3,
...
} amc520_syncctrl_t;
```

Usage:

```
      int fd;
      amc520_adcstatus_t syncctrl;
      ioctl( fd, AMC520_IOC_GET_SYNCCTRL, &syncctrl );
      ioctl( fd, AMC520_IOC_SET_SYNCCTRL, &syncctrl );
```

These calls get/set the ADC synchronizer control parameters.

9.13 AMC520_IOC_GET/SET_DACCTRL

```
#define AMC520_DAC_GEN_ZEROS           0x0
#define AMC520_DAC_GEN_ONES            0x1
#define AMC520_DAC_GEN_MID             0x2
#define AMC520_DAC_GEN_RAMP            0x3
#define AMC520_DAC_GEN_FSDIV2          0x4
#define AMC520_DAC_GEN_FSDIV4          0x5
#define AMC520_DAC_GEN_FSDIV8          0x6
#define AMC520_DAC_GEN_FSDIV16         0x7
#define AMC520_DAC_GEN_FIXED_DATA      0x08
#define AMC520_DAC_GEN_ADC0            0x10
#define AMC520_DAC_GEN_ADC1            0x11
#define AMC520_DAC_GEN_ADC2            0x12
#define AMC520_DAC_GEN_ADC3            0x13
#define AMC520_DAC_GEN_ADC4            0x14
#define AMC520_DAC_GEN_ADC5            0x15
#define AMC520_DAC_GEN_ADC6            0x16
#define AMC520_DAC_GEN_ADC7            0x17
#define AMC520_DAC_GEN_ADC8            0x18
#define AMC520_DAC_GEN_ADC9            0x19

#define AMC520_DAC_BINARY              0
#define AMC520_DAC_TWOS_COMPLEMENT     1

typedef struct {
        unsigned int            reset       : 1;
        unsigned int            xor_en      : 1;
        unsigned int            torb        : 1;
        unsigned int            pd          : 1;
        unsigned int            toggle      : 1; // read-only
        unsigned int            dacinok     : 1; // read-only
        unsigned int            daclocked   : 1; // read-only
        unsigned                gen1        : 5;
        unsigned                fixed_data1 : 16;
        unsigned                gen0        : 5;
        unsigned                fixed_data0 : 16;
} amc520_dacctrl_t;
```

Usage:

```
      int fd;
      amc520_daccommon_t dacctrl;
      ioctl( fd, AMC520_IOC_GET_DACCTRL, &dacctrl );
      ioctl( fd, AMC520_IOC_SET_DACCTRL, &dacctrl );
```

These calls get/set the DAC control parameters.

9.14 AMC520_IOC_GET/SET_DACDELAY

```
typedef struct {
        unsigned                tap : 5;
} amc520_delay_t;
```

Usage:

```
      int fd;
      amc520_delay_t dacdelay;
      ioctl( fd, AMC520_IOC_GET_DACDELAY, &dacdelay );
      ioctl( fd, AMC520_IOC_SET_DACDELAY, &dacdelay );
```

These calls get/set the DAC IODELAY tap setting which can skew the DAC clock relative to the DAC data as it leaves the FPGA if necessary for optimal setup/hold time.

9.15 AMC520_IOC_GET/SET_SYNCDELAY

```
typedef struct {
        unsigned                tap : 5;
} amc520_delay_t;
```

Usage:

```
      int fd;
      amc520_delay_t syncdelay;
      ioctl( fd, AMC520_IOC_GET_SYNCDELAY, &syncdelay );
      ioctl( fd, AMC520_IOC_SET_SYNCDELAY, &syncdelay );
```

These calls get/set the ADC Synchronizer IODELAY tap setting which can skew the ADC Sync pulses relative to the ADC global clock as the pulses leave the FPGA if necessary for optimal setup/hold time.

9.16 AMC520_IOC_GET/SET_ADCDELAY

```
typedef struct {
        unsigned int     chip_idx;              // selects chip for get/set

        unsigned         tap : 5;
} amc520_adcdelay_t;
```

Usage:

```
      int fd;
      amc520_adcdelay_t adcdelay;
      ioctl( fd, AMC520_IOC_GET_ADCDELAY, &adcdelay );
      ioctl( fd, AMC520_IOC_SET_ADCDELAY, &adcdelay );
```

These calls get/set the ADC Data IODELAY tap setting which can skew the ADC Data relative to the ADC channel clock as the data enters the FPGA if necessary for optimal setup/hold time.

9.17 AMC520_IOC_FLASH_OP

```
typedef enum {
        AMC520_FLASH_READ = 0,
        AMC520_FLASH_WRITE,
        AMC520_FLASH_RECONFIG
} amc520_flash_action_t;

typedef struct {
        unsigned int            address;
        unsigned short          data;
        amc004_flash_action_t   action;
} amc520_flash_op_t;
```

Usage:

```
      int fd;
      amc520_flash_op_t flash_op;

      flash_op.address = 0;
      flash_op.data = 0;
      flash_op.action = AMC520_FLASH_READ;
      ioctl( fd, AMC520_IOC_FLASH_OP, &flash_op );
```

This call translates directly to FPGA BPI flash bus transactions for READ and WRITE. When the RECONFIG action is specified the FPGA will reload the configuration data out of the flash. Please refer to the provided device driver and tool application code for a complete example of how to use this ioctl.

10 Appendix C: AMC520 Card-edge Pin-out

AMC Finger	Net	AMC Finger	Net	AMC Finger	Net	AMC Finger	Net	AMC Finger	Net
1	GND	35	AMC/TX3+	69	AMC/RX7-	103	AMC/TX10+	137	GND
2	AMC+12V	36	AMC/TX3-	70	GND	104	GND	138	CLKD-
3	*AMCPS1	37	GND	71	AMCSDA	105	AMC/RX11-	139	CLKD+
4	AMCMP	38	AMC/RX3+	72	AMC+12V	106	AMC/RX11+	140	GND
5	AMCGA0	39	AMC/RX3-	73	GND	107	GND	141	TRIGSTART-
6	n.c.	40	GND	74	CLKA+	108	AMC/TX11-	142	TRIGSTART+
7	GND	41	*AMCENABLE	75	CLKA-	109	AMC/TX11+	143	GND
8	n.c.	42	AMC+12V	76	GND	110	GND	144	TRIGEND-
9	AMC+12V	43	GND	77	CLKB+	111	AMC/RX12-	145	TRIGEND+
10	GND	44	AMC/TX4+	78	CLKB-	112	AMC/RX12+	146	GND
11	AMC/TX0+	45	AMC/TX4-	79	GND	113	GND	147	n.c.
12	AMC/TX0-	46	GND	80	PCI-E/CLK+	114	AMC/TX12-	148	n.c.
13	GND	47	AMC/RX4+	81	PCI-E/CLK-	115	AMC/TX12+	149	GND
14	AMC/RX0+	48	AMC/RX4-	82	GND	116	GND	150	n.c.
15	AMC/RX0-	49	GND	83	*AMCPS0	117	AMC/RX13-	151	n.c.
16	GND	50	AMC/TX5+	84	AMC+12V	118	AMC/RX13+	152	GND
17	AMCGA1	51	AMC/TX5-	85	GND	119	GND	153	n.c.
18	AMC+12V	52	GND	86	GND	120	AMC/TX13-	154	n.c.
19	GND	53	AMC/RX5+	87	AMC/RX8-	121	AMC/TX13+	155	GND
20	AMC/TX1+	54	AMC/RX5-	88	AMC/RX8+	122	GND	156	n.c.
21	AMC/TX1-	55	GND	89	GND	123	AMC/RX14-	157	n.c.
22	GND	56	AMCSCL	90	AMC/TX8-	124	AMC/RX14+	158	GND
23	AMC/RX1+	57	AMC+12V	91	AMC/TX8+	125	GND	159	n.c.
24	AMC/RX1-	58	GND	92	GND	126	AMC/TX14-	160	n.c.
25	GND	59	AMC/TX6+	93	AMC/RX9-	127	AMC/TX14+	161	GND
26	AMCGA2	60	AMC/TX6-	94	AMC/RX9+	128	GND	162	n.c.
27	AMC+12V	61	GND	95	GND	129	AMC/RX15-	163	n.c.
28	GND	62	AMC/RX6+	96	AMC/TX9-	130	AMC/RX15+	164	GND
29	AMC/TX2+	63	AMC/RX6-	97	AMC/TX9+	131	GND	165	AMCTCLK
30	AMC/TX2-	64	GND	98	GND	132	AMC/TX15-	166	AMCTMS
31	GND	65	AMC/TX7+	99	AMC/RX10-	133	AMC/TX15+	167	*AMCTRST
32	AMC/RX2+	66	AMC/TX7-	100	AMC/RX10+	134	GND	168	AMCTDO
33	AMC/RX2-	67	GND	101	GND	135	CLKC-	169	AMCTDI
34	GND	68	AMC/RX7+	102	AMC/TX10-	136	CLKC+	170	GND

Table 23: AMC520 card-edge pin-out

NOTE: Signals shown in Yellow connect to the FPGA. Signals shown in Blue connect to other circuits on the AMC and not the FPGA. Signals shown in plum connect to the FPGA after going through intermediate circuits (i.e. M-LVDS transceivers or JTAG routing).

11 Appendix D: AMC520 RTM Pin-out

Row	GndF	F	E	GndD	D	C	GndB	B	A
10	GND	n.c.	n.c.	GND	n.c.	n.c.	GND	n.c.	n.c.
9		CLK1-	CLK1+		n.c.	n.c.		CLK0-	CLK0+
8		n.c.	n.c.		CLK2-	CLK2+		n.c.	n.c.
7		n.c.	n.c.		n.c.	n.c.		n.c.	n.c.
6		D11-	D11+		D10-	D10+		D09-	D09+
5		D08-	D08+		D07-	D07+		D06-	D06+
4		D05-	D05+		D04-	D04+		D03-	D03+
3		D02-	D02+		D01-	D01+		D00-	D00+
2		n.c.	n.c.		SCL_RTM_L	RTM_MP		+12VRTM	+12VRTM
1		n.c.	n.c.		SDA_RTM_L	*RTM_PS		+12VRTM	+12VRTM

Table 24: AMC520 J30 Pin-out

Row	GndF	F	E	GndD	D	C	GndB	B	A
10	GND	CH0_PA-	CH0_PA+	GND	GND	GND	GND	CH0_TF-	CH0_TF+
9		CH1_TF-	CH1_TF+		OUTI-/0	OUTI+/0		CH1_PA-	CH1_PA+
8		CH2_PA-	CH2_PA+		GND	GND		CH2_TF-	CH2_TF+
7		CH3_TF-	CH3_TF+		OUTQ-/0	OUTQ+/0		CH3_PA-	CH3_PA+
6		CH4_PA-	CH4_PA+		GND	GND		CH4_TF-	CH4_TF+
5		CH5_TF-	CH5_TF+		n.c.	n.c.		CH5_PA-	CH5_PA+
4		CH6_PA-	CH6_PA+		n.c.	n.c.		CH6_TF-	CH6_TF+
3		CH7_TF-	CH7_TF+		n.c.	n.c.		CH7_PA-	CH7_PA+
2		CH8_PA-	CH8_PA+		n.c.	n.c.		CH8_TF-	CH8_TF+
1		CH9_TF-	CH9_TF+		n.c.	n.c.		CH9_PA-	CH9_PA+

Table 25: AMC520 J31 Pin-out

NOTE: Signals shown in Yellow connect to the FPGA. Signals shown in Blue connect to other circuits on the AMC and not the FPGA. Signals shown in plum connect to the FPGA after going through intermediate clocking circuits (i.e. clock distribution).